VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vybrané aplikace metody spektrálních rozkladů.

Selected applications of spectral decomposition methods.

2016 Bc. Michal Zich

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Michal Zich**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Vybrané aplikace metody spektrálních rozkladů
Selected Applications of Spectral Decomposition Methods

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem této práce je prostudovat metody, které využívají spektrálních rozkladů různých matic a použít je v odlišných oblastech. Součástí práce je implementace vybraných metod.

1. Prostudujte teorii k problematice spektrálních rozkladů.
2. Vytvořte přehled těchto metod v různých oblastech.
3. Implementujte několik vybraných metod.
4. Vyberte a popište různě rozsáhlé datové kolekce, nad kterými aplikujete vybrané metody.
5. Porovnejte a analyzujte získané výsledky a vhodně je reprezentujte.

Seznam doporučené odborné literatury:

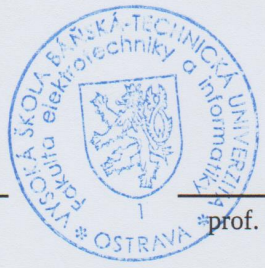
[1] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets. Cambridge University Press.
[2] Ding, C., & He, X. (2004, July). Linearized cluster assignment via spectral ordering. In Proceedings of the twenty-first international conference on Machine learning (p. 30). ACM.
[3] Jia, H., Ding, S., Xu, X., & Nie, R. (2014). The latest research progress on spectral clustering. Neural Computing and Applications, 24(7-8), 1477-1486.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Pavla Dráždilová, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 21. dubna 2016

Michael Zitka

.....

Mé poděkování patří paní Mgr. Pavle Dráždilové, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnovala. Dále bych rád poděkoval panu doc. Mgr. Miloši Kudělkovi, Ph.D. za poskytnutí datových kolekcí k experimentům.

Abstrakt

Tato diplomová práce se zabývá algoritmy využívajících spektrálních rozkladů Laplaceových matic. Cílem práce je prostudovat, naimplementovat a použít tyto algoritmy v odlišných oblastech. Vybrány byly tři spektrálně shlukovací algoritmy, které byly použity pro analýzu různě rozsáhlých datových kolekcí obsahující neorientované vážené grafy a pro segmentaci obrazu. Výsledky shlukové analýzy dat nad kolekcemi grafů byly vyhodnocovány metodami měřící kvalitu shlukování a vizualizovány programem Gephi.

Klíčová slova: diplomová práce, shlukování, shluk, spektrální shlukování, vlastní čísla, vlastní vektory, Alglib, Math.NET Numerics, C#, WPF, analýza dat, k-means, modularita, konduktance, cutratio, silhouette index, dunnův index, teorie grafů, Gephi, programovací jazyk R, segmentace obrazu

Abstract

The master thesis deals with the algorithms that use spectral decompositions of Laplacian matrices. The aim of the thesis is to examine, implement and use these algorithms in various areas. Three spectral clustering algorithms were chosen and used for the analysis of different datasets containing undirected weighted graphs as well as for the image segmentation. The results of clustering analysis of graphs datasets were evaluated by methods measuring the quality of clustering and they were also visualised by Gephi software.

Key Words: master thesis, clustering, cluster, spectral clustering, eigenvalues, eigenvectors, Alglib, Math.NET Numerics, C#, WPF, data analysis, k-means, modularity, conductance, cutratio, silhouette index, dunn index, graph theory, Gephi, programming language R, image segmentation

Obsah

Seznam použitých zkratek a symbolů

Bajt (byte)	–	jednotka množství dat
BitMapa	–	Rastrový obrázek, resp. jeho reprezentace v paměti počítače jako pole pixelů
CSR	–	Compressed Sparse Row formát pro ukládání řídkých matic
Dataset	–	Soubor s daty
Deg(v)	–	Stupeň vrcholu grafu
DOT formát	–	Graph description language
EVD	–	Eigenproblems - problémy vlastních vektorů a vlastních čísel
MB	–	Megabajt - jednotka množství dat
PCA	–	Principal Component Analysis
PDF	–	Portable Document Format
PostScript	–	Programovací jazyk určený ke grafickému popisu tisknutelných dokumentů
PNG	–	Portable Network Graphics
R	–	Statistický programovací jazyk R
SVD	–	Singular Value Decomposition
SVG	–	Scalable Vector Graphics
TFS	–	Team Foundation Server
WPF	–	Windows Presentation Foundation
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V současné době je téma velkých a rozsáhlých dat na každodenním pořádku. Organizace získávají obrovské množství dat, která se snaží nějakým způsobem zpracovávat a získávat z nich informace, které mohou vést k budoucím ziskům či ušetření nákladů. S nástupem internetu věcí bude dat čím dál tím více a je třeba nalézt metody, které nám pomohou tato data zpracovat a vyčíst z nich informace, které se dají použít v náš prospěch.

Tématem této diplomové práce je použití metod využívající spektrálních rozkladů Laplaceových matic, především spektrální shlukování. Shlukování je důležitá součást z oblasti dolování dat. Účelem shlukování je rozdělit datovou kolekci do skupin s tím, že body z jedné skupiny jsou si vzájemně podobné, zatímco s prvky z jiných skupin si podobné nejsou. Tradiční metody, jako jsou k-means nebo hierarchické shlukování, jsou jednoduché, avšak neposkytují dostatečné možnosti pro komplexní datové struktury. Spektrální shlukování vzbudilo rozsáhlou pozornost v akademické obci v posledních letech nejen díky pevné teoretické základně, ale i díky kvalitním výsledkům shlukování. Spektrální shlukování se stalo v posledních letech jednou z nejpopulárnějších moderních shlukovacích metod [1].

Ve druhé kapitole, která převážně obsahuje matematické koncepty používané u spektrálního shlukování, se nejprve zaměříme na teorii grafů. Spektrální shlukování pracuje s objekty jako s vrcholy grafu a hranami reprezentujícími relace mezi objekty. Jednou z nejdůležitějších částí sekce jsou Laplaceovy matice, se kterými spektrální shlukování pracuje a ze kterých se vypočítávají vlastní čísla a vlastní vektory. Dále bude představen základní shlukovací algoritmus k-means, který je použit v rámci algoritmů spektrálního shlukování.

Samotné algoritmy spektrálního shlukování budou popsány ve třetí sekci. V této části se také zaměříme na problémy řezu grafů, kde uvidíme, že rozdílné řezy vedou k rozdílným výsledkům.

Ve 4. sekci představíme dva typy metod, které měří kvalitu shlukování. První typ pracuje se vzdálenostmi mezi prvky, zde patří silhouette index a dunnův index. Druhý typ měří kvalitu shlukování nad grafy a patří sem modularita, konduktance a cut ratio.

Pátá kapitola je zaměřena na implementaci. Budou zde představeny desktopové aplikace, které vznikly za použití .NET frameworku, jeho technologie WPF a programovacího jazyka C#. Následovat bude představení knihoven třetích stran, které budou použity pro matematické výpočty. Jednodušší představu o částech aplikace nám poskytne diagram komponent. Poté představíme různě rozsáhlé datové kolekce, se kterými se v této diplomové práci bude pracovat.

Šestá část nejprve představí vizualizační nástroje, pomocí kterých budeme po analýze spektrálními algoritmy vizualizovat rozdělení vrcholů do shluků. Pro kontrolu výsledků bude sloužit statistický programovací jazyk R. Hlavní částí této kapitoly jsou experimenty, které se dělí do dvou částí. První část pojednává o experimentech nad grafovými datovými kolekcemi, druhá o segmentaci obrazu za použití algoritmů využívající spektrálních rozkladů.

V poslední kapitole této diplomové práce budou, mimo jiné, shrnuty jak výsledky experimentů, tak osobní přínosy plynoucí ze zpracování diplomové práce.

2 Teoretické základy a pojmy

V této části si představíme některé základní pojmy, které budeme potřebovat pro práci s metodami využívajících spektrálních rozkladů.

Podíváme se na problém vlastních čísel a vlastních vektorů, ale hlavní částí této kapitoly budou grafy. Představíme hlavní koncept grafů a budeme mluvit o maticích podobnosti, které popisují párové podobnosti mezi daty daného datového souboru. Budeme definovat Laplaceovy matice a jejich vlastnosti. Nakonec představíme problém řezu grafu (cut problem) reprezentující grafovou verzi shlukovacího problému. Řešení problému řezu grafu nás vede k spektrálním rozkladům matic.

Pokračovat budeme základním, nespektrálním shlukovacím algoritmem *k-means*, který budeme používat pro detekci shluků v redukovaném prostoru.

2.1 Grafy

Zde si představíme základní pojmy z teorie grafů a poté uvedeme speciální typy grafů, které budeme potřebovat. V neposlední řadě budeme mluvit o Laplaceových maticích a řezech grafu, vedoucích k užití spektrálních rozkladů.

2.1.1 Notace

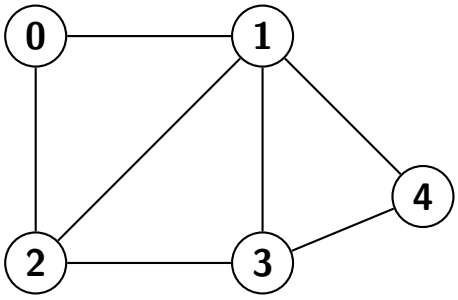
Definice 1 *Graf G = (V,E) je dán množinou vrcholů V = {v_1, v_2,..., v_n} a množinou hran E = {e_1, e_2,...,e_m}, kde $e_i = \{u, v\}; u, v \in V$. Pokud není řečeno jinak, považujme graf za neorientovaný. Hrany mohou mít váhu w(u,v), která může představovat podobnost mezi vrcholy. Předpokládejme, že váhy budou nabývat pouze nezáporných hodnot $w_{ij} > 0$. Pokud nebude uvedeno jinak, budeme počítat u neohodnocených hran s váhou $w_{ij} = 1$.*

Nechť A je matice sousednosti grafu G:

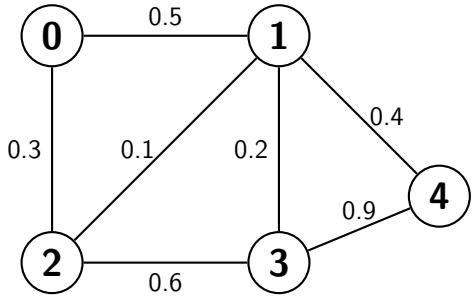
$$A = (a_{ij})_{i,j=1,2,\ldots,n}. \tag{1}$$

Pokud nabývá prvek $a_{ij} = 0$, vrcholy v_i a v_j nejsou spojené hranou. Považujeme-li graf G za neorientovaný, platí $a_{ij} = a_{ji}$. Pro vážený graf se jako váha považuje podobnost mezi vrcholy $w_{ij} = sim(v_i, v_j)$, $w_{ij} = 0$ značí žádnou podobnost.

Jak vypadá neorientovaný, neválený graf lze vidět na obrázku 1, jak jeho neválená matice sousednosti A potom na obrázku 3. Neorientovaný graf s ohodnocenými hranami můžeme spatřit na obrázku 2 a jeho váženou matici sousednosti W na obrázku4.



Obrázek 1: Neorientovaný nevážený graf



Obrázek 2: Neorientovaný vážený graf

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Obrázek 3: Matice sousednosti A

$$W = \begin{bmatrix} 0 & 0.5 & 0.3 & 0 & 0 \\ 0.5 & 0 & 0.1 & 0.2 & 0.4 \\ 0.3 & 0.1 & 0 & 0.6 & 0 \\ 0 & 0.2 & 0.6 & 0 & 0.9 \\ 0 & 0.4 & 0 & 0.9 & 0 \end{bmatrix}$$

Obrázek 4: Matice vah W

Dále budeme definovat diagonální matici $D = \{d_1, d_2, ..., d_n\}$, kde $d_1, d_2, ..., d_n$ jsou prvky na hlavní diagonále:

$$D = \begin{bmatrix} d_1 & 0 & \cdots & 0 & 0 \\ 0 & d_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & d_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & d_n \end{bmatrix}. \tag{2}$$

Hodnota d_i odpovídá v neohodnoceném grafu stupni vrcholu:

$$d_i = \sum_{i=1}^{n} a_{ij},$$

v ohodnoceném grafu pak:

$$d_i = \sum_{i=1}^{n} w_{ij}.$$

Dále budeme používat značení, že $i \in C \Leftrightarrow v_i \in C$ a zavedeme hodnotu řezu:

$$W(B,C) = \sum_{i \in B, j \in C}^{n} w_{ij}$$

pro dvě libovolné množiny $B, C \subseteq V$. Pro podgraf G_B grafu G je definován komplement $G_{\overline{B}}$, kde $B \cup \overline{B} = V$. Později se budeme zabývat kvalitou rozdělení vrcholů grafu. Jednou z cest je použít počet vrcholů v B (značíme $|B|$). Další cestou je suma vah všech hran spojených s vrcholy v B, značíme $vol(B) = \sum_{\substack{v_i \in B \\ v_j \in \overline{B}}} w_{ij}$.

Definice 2 *Nechť G je graf a $B \subseteq V$. Řekněme, že B je souvislá, pokud existuje cesta mezi jakýchkoliv dvěma vrcholy v B.*

Definice 3 *Nechť $B \subseteq V$. Definujme indikátorový vektor množiny B jako $\mathbb{1}_B = (f_1, ..., f_{|V|})$, kde*

$$f_i = \begin{cases} 1 & \textit{pokud } v_i \in B \\ 0 & \textit{pokud } v_i \notin B \end{cases}.$$

2.2 Matice podobnosti

Standardní matice sousednosti A zachycuje pouze zda spolu vrcholy sousedí (1), nebo nesousedí (0). V ohodnoceném případě používáme různé druhy podobnosti mezi v_i a v_j.

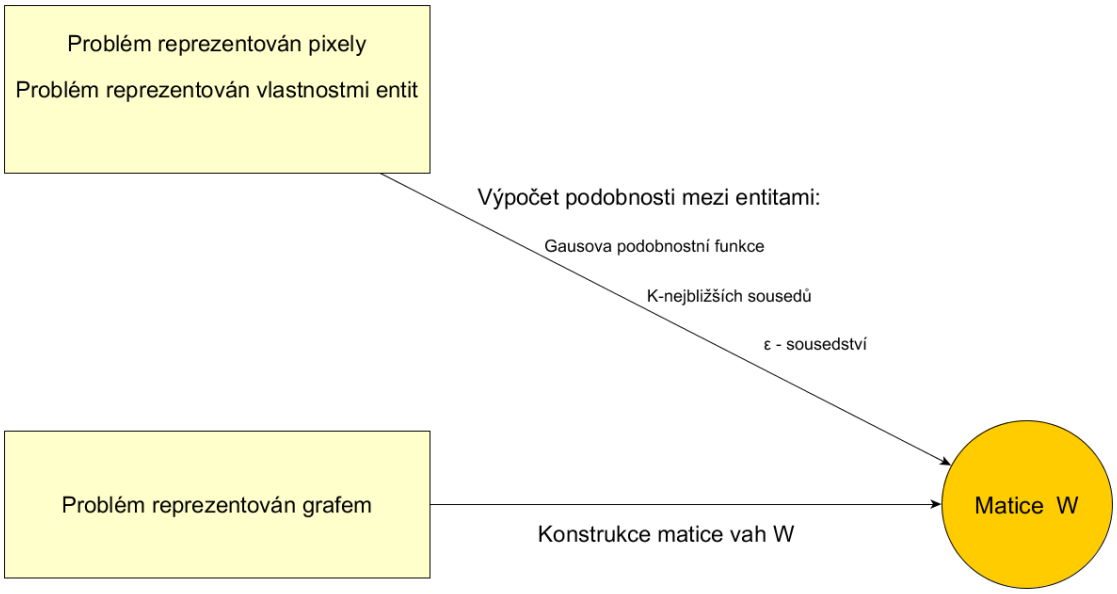
Definice 4 *Nechť $P = \{x_1, ..., x_n\} \in \mathbb{R}^m$ je daná množina dat a máme hodnoty s_{ij} popisující podobnost x_i a x_j pro všechna tato data. Zkonstruovaný graf G = (V,E) s použitím prvků jako vrcholů a ohodnocením každé hrany s odpovídající podobností, t.j. $w_{ij} = s_{ij}$. Poté je G nazýván graf podobnosti (matice podobnosti) této množiny dat.*

Na následujícím obrázků 5 si lze udělat lepší představu o různých typech dat s různými způsoby reprezentací.

Pracujeme-li s takovými datovými kolekcemi, ve kterých jsou problémy již reprezentovány grafy, stačí použít jako matici podobnosti váženou matici sousednosti W, nebo neváženou matici sousednosti A, protože podobnost mezi vrcholy je v nich již uložena. U druhého typu dat je třeba vypočítat podobnost mezi datovými prvky.

Poněvadž pro n prvků budou matice podobnosti matice nxn, a protože n je obvykle velké číslo, potřebujeme zvážit technologické limity pamětí počítačů. V případě nízké podobnosti prvků, nebo v případě, že dva vrcholy mezi sebou nemají hranu, budeme ukládat do matice W nuly. Tento přístup umožňuje pracovat s maticí W jako s řídkou maticí.

Cílem této kapitoly je projednat rozdílné přístupy jak mohou být tyto řídké matice podobnosti W zkonstruovány.



Obrázek 5: Možnosti konstrukce matice W

2.2.1 Kompletní matice podobnosti

Nejjednodušší cesta, jak zkonstruovat matici podobnosti, je jednoduché spojení všech vrcholů x_i a x_j za předpokladu $s_{ij} \neq 0$. Mějme na paměti, že matice podobnosti by měla modelovat lokální sousedství, standardní přístup pro kompletní grafy je například Gaussova podobnostní funkce:

$$s_{ij} = exp\left(-\frac{d(x_i, x_j)^2}{2\sigma^2}\right), \tag{3}$$

kde σ je parametr kontrolující velikost tohoto sousedství. Metrika d může být, například, euklidovská metrika. Očividná nevýhoda tohoto typu podobnostní matice je to, že postrádá řídkou strukturu. U větších datových kolekcí můžeme přistoupit k možnosti zanedbat některá velmi malá čísla znázorňující velmi nízkou podobnost datových bodů a místo této nízké hodnoty vkládat do matice nuly. Díky tomuto dostaneme řídkou strukturu matice. Této problematice tzv. *tresholdu* se budeme věnovat v kapitole o implementaci matice podobnosti pro segmentaci obrazu.

2.2.2 Graf k-nejbližších sousedů (k-Nearest Neighbors)

Dalším přístupem je typ grafu k-nejbližších sousedů. Spojíme v_i s v_j, pokud v_j je mezi k nejbližšími sousedy v_i, kde k je fixní číslo. Tato definice vede k orientovanému grafu, protože vztah sousedství není symetrický. Převedení tohoto grafu na neorientovaný podobnostní graf, můžeme použít, mimo jiné, jednu z těchto dvou metod:

- *Normální k-nejbližší sousedi*: Spojíme vrcholy pokud jeden z nich je mezi k-nejbližšími sousedy toho druhého.

- *Vzájemní k-nejbližší sousedi*: Spojíme vrcholy pokud v_i je mezi k-nejbližšími sousedy v_j a v_j je mezi k-nejbližšími sousedy v_i.

Pro oba typy použijeme podobnost v_i a v_j k ohodnocení hrany.

2.2.3 Graf ε-sousedství (ε-Neighborhood)

Poslední typ, na který se podíváme, je graf typu ε-sousedství. Zde jednoduše spojíme v_i a v_j pokud $d(x_i, x_j) < \varepsilon$ pro fixní práh $\varepsilon > 0$. Vzhledem k tomu, že vzdálenosti mezi spojenými body jsou zhruba ve stejném rozsahu, nejvýše ε, váhy hran neobsahují mnoho informací o daném grafu. Z tohoto důvodu je graf ε-sousedství považován za nevážený [1].

2.3 Vlastní čísla a vlastní vektory

Definice 5 *Nechť $A \in L(V)$ je lineární transformace definovaná na vektorovém prostoru V. Jestliže existuje nenulový vektor $\boldsymbol{e} \in \mathbb{V}$ a skalár λ tak, že*

$$Ae = \lambda e,$$

*pak se λ nazývá **vlastní číslo** transformace A, $\boldsymbol{e}$ se nazývá **vlastní vektor** příslušný k vlastnímu číslu λ [29].*

2.4 Laplaceovy matice a jejich vlastnosti

Laplaceovy matice určitým způsobem reprezentují grafy. Teorie spektrálních rozkladů studuje tyto matice, protože v sobě obsahují mnoho užitečných informací o grafech.

Považujme G jako neorientovaný a vážený graf, W jako ohodnocenou matici sousednosti a D jako diagonální matici definovanou v předchozí kapitole 2.1.

V této části se budeme zabývat nejdůležitějšími vlastnostmi těchto matic, které budeme dále využívat v teorii spektrálních rozkladů.

Mezi první průkopníky spektrální teorie Laplaceových matic patřil i český vědec Prof. RNDr. Miroslav Fiedler, DrSc.[1], který v roce 1973 zavedl pojem *algebraická souvislost* a objevil, že pro dělení grafu na dvě partity je vhodné využít vlastností vlastního vektoru příslušného druhému nejmenšímu vlastnímu číslu [3]. Tento vektor se nazývá Fiedlerův vektor.

[1] pan profesor, bohužel, loni zemřel (7. dubna 1926 - 20. listopadu 2015).

2.4.1 Nenormovaná Laplaceova matice

Nenormovaná Laplaceova matice L je definována jako rozdíl diagonální matice D a vážené matice sousednosti W

$$L = D - W. \tag{4}$$

Mezi její nejpodstatnější vlastnosti patří, že je pozitivně semidefinitní, symetrická a singulární. Nejmenší vlastní číslo nenormované Laplaceovy matice je rovno 0 a jeho násobnost je rovna počtu souvislých komponent grafu. Pro vlastní vektory příslušné nulovým vlastním číslům nenormované Laplaceovy matice pak platí několik následujících vlastností. Tyto vlastnosti, věty i důkazy jsou přejaty z [1].

Věta 1 *Nechť L je Laplaceova matice. Poté máme následující vlastnosti:*

1. *Pro každý vektor $f \in \mathbb{R}^n$ máme*

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

2. *L je symetrická a pozitivně semidefinitní.*

3. *Nejmenší vlastní číslo matice L je 0, příslušný vlastní vektor je konstantní jednotkový vektor $\mathbb{1}$.*

4. *L má n nezáporných, reálných vlastních čísel $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.*

Důkaz. Nyní se podívejme na tato prohlášení jednotlivě:

1. Užitím definice L a stupně vrcholu d_i, dostaneme

$$\begin{aligned} f'Lf &= f'(D-W)f \\ &= f'Df - f'Wf \\ &= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2\right) \\ &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2. \end{aligned}$$

Toto ukazuje, že $f'Lf \geq 0$ pro všechna $f \in \mathbb{R}^n$.

2. W je symetrická, protože předpokládáme graf jako neorientovaný. Diagonální matice D je očividně také symetrická, a proto je L taktéž symetrická.

   Pozitivní semidefinitnost vyplývá z $f'Lf \geq 0$ pro všechna $f \in \mathbb{R}^n$, což jsme právě dokázali.

3. Nyní chceme ověřit, zda $L\mathbb{1} = 0$. Z definice o Laplalaceově matici plyne, že $D\mathbb{1} = W\mathbb{1}$. Obě $D\mathbb{1}$ a $W\mathbb{1}$ jsou nx1 vektory obsahující řádek součtu D respektive W. Součet i-tého řádku z matice W je přesně to, co jsme definovali jako d_i, což je také součtem i-tého řádku z matice D, protože D je diagonální matice. To ukazuje, že oba vektory jsou stejné, což dokazuje že $L\mathbb{1} = 0$.

4. Ukázali jsme, že L je symetrická a pozitivně definitní. Tudíž všechna vlastní čísla jsou reálná a nezáporná.

■

Důležitou vlastností pro spektrální shlukování je zejména následující výsledek.

Věta 2 *Nechť G je neorientovaný, vážený graf. Poté geometrická násobnost k vlastního čísla 0 matice L se rovná počtu souvislých komponent v grafu. Vlastní prostor tohoto vlastního čísla je rozložen pomocí indikátorových vektorů těchto komponent.*

Důkaz. Začneme tím, že se podíváme na souvislý graf, jehož $k = 1$. Nechť f je vlastní vektor s vlastním číslem 0, poté, podle 2, máme

$$0 = f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Jak již bylo zmíněno v důkazu první vlastnosti v 1, součet se může rovnat nule jen v případě, že všechny sumy $w_{ij}(f_i - f_j)^2$ zmizí. Vzhledem k tomu, že máme $w_{ij} > 0$ pro dva spojené vrcholy v_i a v_j, dojdeme k závěru, že $f_i = f_j$. Aby byl graf souvislý, potřebuje f být konstantní pro všechny vrcholy v grafu. Vzhledem k tomu, že předpokládáme souvislý graf, f musí být konstantní v celém grafu. Samozřejmě každá násobnost vlastního vektoru je vlastní vektor, ale protože s ním zacházíme jako s jedním, vidíme, že $f = \mathbb{1}$ je jediný vlastní vektor s vlastním číslem 0. Je tedy zřejmé, že tohle je indikátorový vektor pro spojené komponenty.

Nyní se podíváme na libovolné k, s tím, že máme k souvislých komponent. Očividně lze změnit pořadí vrcholů aniž bychom změnili graf, a proto jejich řazení pomocí souvislých komponent, do

kterých patří, můžeme dosáhnout toho, že matice sousednosti W má diagonální tvar. Pak tedy získáme takovýto tvar matice:

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{bmatrix}. \tag{5}$$

Každý z těchto bloků L_i je sama o sobě Laplaceova matice, a proto reprezentuje graf. Přesněji to reprezentuje i-tou souvislou komponentu podgrafu G. Vzhledem k tomu, že L má diagonální tvar, její spektrum se rovná sjednocení spekter L_i a dostaneme odpovídající vlastní vektory vyplněním všech dalších položek nulami. V prvé řadě to ukazuje, že každá L_i má konstantní vektor $\mathbb{1}$ jako vlastní vektor s vlastním číslem 0. Proto se počet vlastních čísel 0 z L rovná počtu souvislých komponent a to odpovídá tomu, že vlastní vektory jsou právě jejich indikátorové vektory [1].

■

2.4.2 Normované Laplaceovy matice a jejich vlastnosti

Nyní nám známá Laplaceova matice je často označována jako *nenormovaná* Laplaceova matice, což naznačuje, že může být v určitém smyslu *normována*. Ve skutečnosti je zde několik cest, jak ji normovat, ale ty obvykle nejsou pojmenovány a jsou pouze označovány jako normované Laplaceovy matice. Nyní si představíme dva typy normovaných Laplaceových matic. Matice L_{sym} a L_{rw} jsou svými vlastnostmi navzájem analogické. Z praktických důvodu je vhodnější použít symetrickou verzi normované Laplaceovy matice, neboť hodnoty vlastních vektorů patří do oboru reálných čísel $\mathbb{R}$, kdežto v případě nesymetrické L_{rw} můžou nabývat prvky vlastních vektorů hodnot z oboru komplexních čísel $\mathbb{C}$.

Definice 6 *Definujeme dvě normalizované Laplaceovy matice:*

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \tag{6}$$

a

$$L_{rw} = D^{-1} L. \tag{7}$$

Pojmenování těchto matic má následující význam: L_{sym} je symetrická matice a L_{rw} souvisí s náhodnou procházkou (random walk). Stejně jak v minulé kapitole se nyní podíváme na několik důležitých vlastností obou Laplaceových matic, které jsou opět převzaty z [1].

Věta 3 *Nechť L_{sym} a L_{rw} jsou výše definovány. Poté máme následující vlastnosti:*

1. *Pro každý vektor $f \in \mathbb{R}^n$ máme*

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 .$$

2. *λ je vlastní číslo L_{rw} s vlastním vektorem u tehdy a jen tehdy když λ je vlastní hodnota L_{sym} s vlastním vektorem $w = D^{\frac{1}{2}} u$.*

3. *λ je vlastní číslo L_{rw} s vlastním vektorem u tehdy a jen tehdy když λ a u řeší všeobecný vlastní problém*

$$Lu = \lambda Du.$$

4. *0 je vlastní číslo L_{rw} s konstantním jednotkovým vektorem $\mathbb{1}$ jako vlastní vektor. 0 je vlastní číslo L_{sym} s vlastním vektorem $D^{\frac{1}{2}} \mathbb{1}$.*

5. *L_{sym} a L_{rw} jsou pozitivně semidefinitní a mají n nezáporných, reálných vlastních čísel 0 $= \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.*

Důkaz. Nyní se znovu podívejme na tato prohlášení jednotlivě:

1. Matice $D^{-\frac{1}{2}}$ je diagonální matice obsahující $\frac{1}{\sqrt{d_i}}$ na i-tém elementu diagonály. Proto $f' D^{-\frac{1}{2}}$ a $D^{-\frac{1}{2}} f$ jsou jen zmenšené verze vektorů f' a f, kde každý člen f_i byl nahrazen $\frac{f_i}{\sqrt{d_i}}$. Substituujeme-li tyto vektory za g' a g, máme stejnou situaci jako v 1.

2. Nechť w je vlastní vektor L_{sym} k vlastnímu číslu λ, tak $L_{sym} = \lambda w$. Vynásobení s $D^{-\frac{1}{2}}$ zleva nám dá $D^{-\frac{1}{2}} L_{sym} w = D^{-\frac{1}{2}} \lambda w$. Tedy

$$D^{-\frac{1}{2}} L_{sym} = D^{-\frac{1}{2}} D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = D^{-1} L D^{-\frac{1}{2}} = L_{rw} D^{-\frac{1}{2}},$$

   nás vede k $L_{rw} D^{-\frac{1}{2}} w = \lambda D^{-\frac{1}{2}} w$, a substituujeme-li $u = D^{-\frac{1}{2}} w$, pak zajistíme, že platí $L_{rw} u = \lambda u$ pro L_{rw}.

3. Počínaje $L_{rw} u = \lambda u$, vynásobíme D na obě strany zleva $D L_{rw} = D D^{-1} L = L$, dostaneme $Lu = \lambda Du$.

4. Z věty 1 víme, že $L\mathbb{1} = 0$. Pak dostaneme $L_{rw}\mathbb{1} = D^{-1} L \mathbb{1} = D^{-1} 0 = 0$, t.j $\mathbb{1}$ je vlastní vektor k vlastnímu číslu 0. Druhý výrok je přímým důsledkem 2.

5. Stejně jako v 1, vlastnosti o L_{sym} vyplývají z 1 a z 2 a tedy i L_{rw} má nezáporné vlastní čísla a je pozitivně semidefinitní [29].

■

2.5 Řez grafu

Nyní hledáme rozklad[2] grafu tak, že každá skupina představuje množinu vrcholů s vysokou podobností k sobě samým, zatímco body z odlišných množin mají velmi nízkou podobnost. Jinými slovy se snažíme najít rozklad grafu takový, kde hrany mezi rozdílnými skupinami mají velmi nízké váhy a hrany uvnitř skupiny mají vysoké váhy.

Definice 7 *Nechť W je vážená matice sousednosti daného grafu a $k \in \mathbb{N}$, poté minimalizace*

$$\mathrm{cut}(B_1, ..., B_k) = \frac{1}{2} \sum_{i=1}^{k} W(B_i, \overline{B_i}),$$

pro $B_1, ..., B_k \subset V$ se nazývá **mincut** problém.

Řešení *mincut* problému, zejména pro $k = 2$, je poměrně snadný problém a může být řešen velmi efektivně [10]. Nicméně v praxi nevede často k uspokojivým rozkladům. Řešení často vede k oddělení samostatného vrcholu od zbytku grafu. Proto chceme pracovat za podmínky, že množiny B_1,...,B_2 budou „přiměřeně velké“. Následující dva problémy řezu patří mezi běžně používané [1].

Definice 8 *Předpokládejme matici sousednosti W jako výše,* RatioCut problém *sestává z minimalizace*

$$\mathrm{RatioCut}(B_1, ..., B_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(B_i, \overline{B_i})}{|B_i|} = \sum_{i=1}^{k} \frac{\mathrm{cut}(B_i, \overline{B_i})}{|B_i|}.$$

Definice 9 *Předpokládejme matici sousednosti W jako výše,* Ncut problém *(zkráceně pro normovaný cut) sestává z minimalizace*

$$\mathrm{Ncut}(B_1, ..., B_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(B_i, \overline{B_i})}{vol(B_i)} = \sum_{i=1}^{k} \frac{\mathrm{cut}(B_i, \overline{B_i})}{vol(B_i)}.$$

Oba tyto problémy se snaží vyvážit velikost skupin B_i, ale každý se toho snaží dosáhnout jinou cestou: *RatioCut* problém bere v úvahu počet vrcholů v každé skupině, zatímco *Ncut* problém pro to používá váhy hran. Zavedením vyvažovacích podmínek se z dříve snadno vyřešitelného *mincut* problému stávají tyto problémy NP-těžkými [6].

2.6 Základní shlukovací algoritmy

Základním účelem shlukování (clustering) je najít shluk (cluster) objektů, které mají navzájem více společných znaků. Hledáme tedy skupinu objektů, které jsou si vzájemně podobné a seskupujeme je do přirozených shluků.

[2] Rozklad je množina neprázdných, disjunktních podgrafů takových, že jejich sjednocení se rovná celému grafu.

Definice 10 *Shlukování (clustering) C je množina všech shluků grafu, tedy* $C = \{C_1, C_2, ..., Ck\}$*, a číslo k počtu shluků. Pokud není uvedeno jinak,* $C_i \cap C_j = \emptyset, \forall i \neq j$*. Shluk* C_i*, který obsahuje pouze jeden vrchol se nazývá singleton [5].*

2.6.1 K-means algoritmus

Shlukovací algoritmus k-means pracuje s datovou kolekcí objektů, které jsou popsány n-dimenzionálními vektory, a pomocí metrik se počítá vzdálenost mezi nimi. Je založen na metodě nejbližších centroidů. Po počáteční náhodné inicializaci středů se centroid přepočítává v několika iteracích. Do shluku jsou přiřazeny jen ty objekty, jejichž vzdálenost k centroidu je nejmenší. Výsledkem je přiřazení každého objektu do jednoho shluku. Tomuto typu shlukování se říká *hard*. Odlišným typem je *soft* shlukování, u kterého může objekt spadat do více shluků (*fuzzy k-means*). Algoritmus končí, jakmile se centroidy přestanou měnit a jejich hodnota zůstává stálá [31].

Algoritmus 1 K-means algoritmus

Vstup: Číslo k počtu shluků.

Výstup: Prvky přidělené do shluků $C_1, ..., C_k$.

1: Vyber k náhodných centroidů.
2: Přiřaď každý objekt do skupiny s nejbližším centroidem.
3: Jakmile jsou všechny objekty zařazeny, přepočti pozice k centroidů.
4: Opakuj kroky 2 a 3 dokud se centroidy nepřestanou měnit.

Hlavními výhodami algoritmu k-means je jednoduchá implementace, rychlost zpracování dat a možnost zpracovávat i velké datové kolekce. Mezi nevýhody se řadí náhodný výběr počtu shluků, citlivost na šum v datech (odlehlé hodnoty), které způsobí vychýlení středu shluku. Dalším problémem je počáteční výběr středů centroidů. Tento výběr probíhá v klasické verzi k-means náhodně, což může způsobit, že se vyberou centroidy, které jsou blízko sebe. Navíc pomocí náhodného výběru je obtížné získat stabilní výsledky měření, protože centroidy mají pokaždé jiné souřadnice a tudíž obdržíme různé výsledky [31].

3 Shlukování pomocí spektrálních rozkladů

Nyní máme všechny potřebné podklady k tomu, abychom mohli začít pracovat s algoritmy využívajících spektrálních rozkladů. Nejprve se zaměříme na problémy řezu grafu, o kterých jsme diskutovali v kapitole 2.5. V této kapitole uvidíme, že různé řezy grafů vedou k různým algoritmům. Posléze budeme mluvit o složitosti algoritmů spektrálního shlukování a také prodiskutujeme některé z případů, ve kterých je možné spektrální shlukování použít.

3.1 Nenormovaný algoritmus

Aby bylo možné algoritmus lépe pochopit, začneme s aproximací řezu *RatioCut* v nejjednodušším případě pro dva shluky. V druhé části představíme myšlenku scénáře pro libovolný počet k shluků.

3.1.1 Aproximace RatioCut pro k = 2

Začněme s případem RatioCutu pro k = 2, protože je jednodušší pochopit tento problém z tohoto nastavení. Naším cílem je vyřešit optimalizační problém:

$$\min_{B \subset V} RatioCut(B, \overline{B}). \tag{8}$$

Je dána podmnožina $B \subseteq V$ a definujeme vektor $f = (f1, ..., fn)' \in \mathbb{R}^n$ s hodnotami:

$$f_i = \begin{cases} \sqrt{\frac{|\overline{B}|}{|B|}} & \text{pokud } v_i \in B \\ -\sqrt{\frac{|B|}{|\overline{B}|}} & \text{pokud } v_i \notin B. \end{cases} \tag{9}$$

Nyní cílová funkce *RatioCutu* může být vhodněji přepsána použitím nenormované Laplaceovy matice. A to díky následujícího výpočtu:

$$\begin{aligned} f'Lf &= \frac{1}{2} \sum_{i,j=1} w_{ij}(f_i - f_j)^2 \\ &= \frac{1}{2} \sum_{i \in B, j \in \overline{B}} w_{ij} \left(\sqrt{\frac{|\overline{B}|}{|B|}} + \sqrt{\frac{|B|}{|\overline{B}|}} \right)^2 + \frac{1}{2} \sum_{i \in \overline{B}, j \in B} w_{ij} \left(-\sqrt{\frac{|\overline{B}|}{|B|}} - \sqrt{\frac{|B|}{|\overline{B}|}} \right)^2 \\ &= cut(B, \overline{B}) \left(\frac{|B|}{|\overline{B}|} + \frac{|B|}{|\overline{B}|} + 2 \right) = cut(B, \overline{B}) \left(\frac{|B| + |\overline{B}|}{|B|} + \frac{|B| + |\overline{B}|}{|\overline{B}|} \right) \\ &= |V| \cdot RatioCut(B, \overline{B}). \end{aligned}$$

Dále máme:

$$\sum_{i=1} f_i = \sum_{i \in B} \sqrt{\frac{|\overline{B}|}{|B|}} - \sum_{i \in \overline{B}} \sqrt{\frac{|B|}{|\overline{B}|}} = |B|\sqrt{\frac{|\overline{B}|}{|B|}} - |\overline{B}|\sqrt{\frac{|\overline{B}|}{|B|}} = 0.$$

Jinými slovy vektor f definován v 9 je ortogonální konstantní jednotkový vektor $\mathbb{1}$. Uvědomme si, že f splňuje

$$||f||^2 = \sum_{i=1} f_i^2 = |B|\frac{|\overline{B}|}{|B|} + |\overline{B}|\frac{|B|}{|\overline{B}|} = |\overline{B}| + |B| = n.$$

Můžeme vidět, že problém minimalizace 8 může být ekvivalentně přepsán jako:

$$\min_{B \subset V} f'Lf \text{ kde } \begin{cases} f \perp \mathbb{1}, \text{jak definováno v 9} \\ ||f|| = \sqrt{n}. \end{cases} \tag{10}$$

Jedná se o diskrétní optimalizační problém, jelikož prvkům vektoru f je dovoleno vzít pouze dvě konkrétní hodnoty a tento problém je stále NP-těžký. Nejvíce vyhovující „uvolnění“ (relaxation) je ve zbavení se diskrétního stavu a místo toho povolit, že f_i může použít libovolnou hodnotu z $\mathbb{R}$. To nás vede k optimalizaci „uvolněného“ (relaxed) problému:

$$\min_{f \in \mathbb{R}^n} f'Lf \text{ kde } \begin{cases} f \perp \mathbb{1} \\ ||f|| = \sqrt{n}. \end{cases} \tag{11}$$

Na základě Rayleigh-Ritz teorému [30] je možno vidět, že řešení tohoto problému je dáno vektorem f, což je vlastní vektor odpovídající druhému nejmenšímu vlastnímu číslu matice L. Tudíž lze aproximovat a minimalizovat *RatioCut* na základě druhého vlastního vektoru matice L. Abychom dosáhli rozdělení grafu, musíme znovu přetransformovat reálná řešení vektoru f „uvolněného“ problému do diskrétního indikátorového vektoru. Nejjednodušší způsob je použít f jako indikátorovou funkci a vybrat:

$$\begin{cases} v_i \in B & \text{pokud} \quad f_i \geq 0 \\ v_i \in \overline{B} & \text{pokud} \quad f_i < 0. \end{cases}$$

Většina spektrálních shlukovacích algoritmů zvažuje souřadnice f_i jako body v $\mathbb{R}$ a shlukuje je do dvou skupin $C, \overline{C}$ pomocí k-means. Poté přeneseme výsledné shlukování do základních datových prvků, které volíme:

$$\begin{cases} v_i \in B & \text{pokud} \quad f_i \in C \\ v_i \in \overline{B} & \text{pokud} \quad f_i \in \overline{C}. \end{cases}$$

Což značí nenormovaný spektrální shlukovací algoritmus pro případ k = 2.

3.1.2 Aproximace RatioCut pro libovolné k

Zjednodušení minimalizace problému *RatioCut* v případě libovolné hodnoty k používá podobné principy jako ten v předchozí kapitole. Je dána množina V, rozdělena do oddílů $B_1, ..., B_k$, a definujeme k indikátorových vektorů $h_j = (h_{1,j}, ..., h_{n,j})$ jakožto

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{|B_j|}} & \text{pokud} \quad v_i \in B_j \\ 0 & \text{pokud} \quad v_i \notin B_j \end{cases} \text{(i = 1,...,n; j = 1,...,k).} \tag{12}$$

Poté zavedeme matici $H \in R^{nxk}$, která obsahuje těchto k indikátorových vektorů jako sloupce. Sloupce jsou samy k sobě ortogonální, t.j. $H'H = I$. Podobný výpočet jako v předchozí sekci můžeme vidět

$$h_i' L h_i = \frac{cut(B_i, \overline{B}_i)}{|B_i|}.$$

A také platí, že:

$$h_i' L h_i = (H'LH)_{ii}.$$

Kombinací těchto dvou faktů dostaneme

$$RatioCut(B_1, ..., B_k) = \sum_{i=1}^{k} h_i' L h_i = \sum_{i=1}^{k} (H'LH)_{ii} = Tr(H'LH),$$

kde Tr označuje stopu matice. Problém minimalizace $RatioCut(B_1, ..., B_k)$ může být přepsán jako

$$\min_{B_1,...,B_k} Tr(H'LH) \text{ kde} \begin{cases} H \text{ jak bylo definováno v 12} \\ H'H = I. \end{cases} . \tag{13}$$

Podobně jako výše, můžeme problém „uvolnit“ (relax), když povolíme hodnotám matice H použít libovolné reálné hodnoty. Poté se zjednodušený problém stane:

$$\min_{H \in R^{nxk}} Tr(H'LH) \text{ kde } H'H = I.$$

Jedná se o standardní formu problému stopové minimalizace a znovu se jedná o verzi Rayleigh-Ritz teorému [30] říkajícího, že řešení je dáno volbou matice H jakožto matice obsahující prvních k vlastních vektorů z matice L jako sloupce. Matice H je v tomto případě matice U použita v nenormovaném spektrálním shlukovacím algoritmu, který je popsán dále. Opět musíme převést reálné hodnoty řešení na „diskrétní“ hodnoty příslušnosti do shluku (partition). Standardní cestou je opět použít k-means algoritmus na sloupce matice U. Toto nás vede k obecnému nenormovaného spektrálně shlukovacímu algoritmu definovaném v 3.1.3 [1].

3.1.3 Nenormovaný spektrální shlukovací algoritmus

Následující algoritmus používá nenormovanou Laplaceovu matici L, což je důvod, proč je algoritmus nazván jako nenormovaný.

Algoritmus 2 Nenormovaný algoritmus

Vstup: Datová kolekce, číslo k počtu shluků

Výstup: Vrcholy grafu přidělené do shluků $C_1, ..., C_k$

1: Sestav váženou matici sousednosti W = $w_{i,j}$ pro i = 1,...,n.
2: Sestav nenormovanou Laplaceovu matici L = D - W.
3: Vypočítej prvních k vlastních vektorů $u_1, ..., u_k$ z L.
4: Nechť $U \in \mathbb{R}^{nxk}$ je matice obsahující vektory $u_1, ..., u_k$ jako sloupce.
5: Pro i = 1,...,n nechť $y_i \in \mathbb{R}^k$ je vektor odpovídající i-tému řádku matice U.
6: Shlukni body $(y_i)_{i=1}^n$ do shluků $C_1, ..., C_k$ za použití k-means algoritmu.

3.2 Normované algoritmy

V předchozí sekci jsme uvedli, že aproximace *RatioCut* vede k algoritmu využívající nenormovanou Laplaceovu matici L. V této kapitole uvedeme aproximaci řezu *Ncut*, jež vede k normovaným Laplaceovým maticím.

3.2.1 Aproximace Ncut pro k = 2

Velmi podobné techniky, které byly použity pro *RatioCut*, mohou být použity i pro odvození normalizovaných spektrálních shlukovacích algoritmů jako zjednodušení minimalizace *Ncut* problému. V případě $k = 2$ definujeme shlukový indikátorový vektor f jako

$$f_i = \begin{cases} \sqrt{\frac{vol(\overline{B})}{vol(B)}} & \text{pokud } v_i \in B \\ -\sqrt{\frac{vol(B)}{vol(\overline{B})}} & \text{pokud } v_i \notin B. \end{cases} \tag{14}$$

Podobně jako u výše zmíněného se dá zjistit, že $(Df)'\mathbb{1} = 0, f'Df = vol(V)$, a $f'Lf = vol(V)Ncut(B, \overline{B})$. Můžeme tedy přepsat problém minimalizace *Ncut* ekvivalentním problémem

$$\min_B f'Lf \text{ kde } \begin{cases} f \text{ jak bylo definováno v 14 a } Df \perp \mathbb{1} \\ f'Df = vol(V). \end{cases} \tag{15}$$

Znovu zjednodušíme problém povolením vzít libovolné reálné hodnoty pro f:

$$\min_{f \in \mathbb{R}^n} f'Lf \text{ kde } \begin{cases} Df \perp \mathbb{1} \\ f'Df = vol(V). \end{cases} \tag{16}$$

Nyní substitujme $g = D^{1/2}f$. Po substituci je problém

$$\min_{g\in\mathbb{R}^n} g'D^{-1/2}LD^{-1/2}g \text{ kde } \begin{cases} g \perp D^{1/2}\mathbb{1} \\ ||g||^2 = vol(V). \end{cases} \tag{17}$$

Lze si všimnout, že $D^{-1/2}LD^{-1/2} = L_{sym}$, $D^{1/2}\mathbb{1}$ je první vlastní vektor matice L_{sym} a $vol(V)$ je konstanta. Z tohoto důvodu je problém 17 ve formě standardního Rayleigh-Ritz teorému [30], a jeho řešení je dáno druhým vlastním vektorem L_{sym}. Substitucí $f = D^{-1/2}g$ a použitím 2.4.2 můžeme vidět, že f je druhý vlastní vektor matice L_{rw}, nebo ekvivalentně generalizovaný vlastní vektor L_{sym}.

3.2.2 Aproximace Ncut pro libovolné k

Pro případ, kde $k > 2$, definujeme indikátorové vektory $h_j = (h_{1,j}, ..., h_{n,j})'$ jako

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{vol(B_j)}} & \text{pokud} v_i \in B_j \\ 0 & \text{pokud} v_i \notin B_j \end{cases} \text{ (i = 1,...,n; j = 1,...,k).} \tag{18}$$

Poté použijeme matici H, která obsahuje těchto k indikátorových vektorů jako sloupce. Všimněme si, že $H'H = I, h_i'Dh_i = 1$ a $h_i'Lh_i = \frac{cut(B_i,\overline{B}_i)}{vol(A_i)}$. Problém minimalizace Ncut lze napsat jako:

$$\min_{A_1,...,A_k} = Tr(H'LH) \text{ kde } \begin{cases} H \text{ jak bylo definováno v 18} \\ H'DH = I. \end{cases}$$

Zjednodušením diskrétní podmínky a substitucí $T = D^{1/2}H$ dostaneme zjednodušený problém

$$\min_{T\in R^{nxk}} = Tr(T'D^{-1/2}LD^{-1/2}T) \text{ kde } T'T = I. \tag{19}$$

Tohle je opět standardní stopová minimalizace problému, který je řešen maticí T, která obsahuje prvních k vlastních vektorů matice L_{sym} jako sloupce. Substitucí $H = D^{-1/2}T$ a použitím 2.4.2 můžeme vidět, že řešení H sestává z prvních k vlastních vektorů matice L_{rw}, nebo prvních k vlastních vektorů L_{sym} [1].

3.2.3 Normované spektrální shlukovací algoritmy

Zjednodušení *Ncut* problému vede k algoritmu, který používá vlastní vektory z matice L_{rw}, proto jej nazýváme normovaný algoritmus. Další algoritmus znázorňuje jinou verzi normovaného spektrálního shlukovacího algoritmu používající normovanou matici L_{sym}, který má jeden krok navíc a to normalizace řádků matice U obsahující indikátorové vlastní vektory jako sloupce.

Algoritmus 3 Normovaný (L_{rw}) algoritmus podle Shi a Malika (2000)

Vstup: Datová kolekce, číslo k počtu shluků

Výstup: Vrcholy grafu přidělené do shluků $C_1, ..., C_k$

1: Sestav váženou matici sousednosti W = $w_{i,j}$ pro i = 1,...,n a diagonální matici D.
2: Sestav nenormovanou Laplaceovu matici L = D - W.
3: Sestav nesymetrickou normovanou Laplaceovu matici $L_{rw} = D^{-1}L$.
4: Vypočítej prvních k vlastních vektorů $u_1, ..., u_k$ z matice L_{rw}.
5: Nechť $U \in \mathbb{R}^{nxk}$ je matice obsahující vektory $u_1, ..., u_k$ jako sloupce.
6: Pro i = 1,...,n nechť $y_i \in \mathbb{R}^k$ je vektor odpovídající i-tému řádku matice U.
7: Shlukni body $(y_i)_{i=1}^n$ do shluků $C_1, ..., C_k$ za použití k-means algoritmu.

Algoritmus 4 Normovaný (L_{sym}) algoritmus podle Ng, Jordana a Weisse (2002)

Vstup: Datová kolekce, číslo k počtu shluků

Výstup: Vrcholy grafu přidělené do shluků $C_1, ..., C_k$

1: Sestav váženou matici sousednosti W = $w_{i,j}$ pro i = 1,...,n a diagonální matici D.
2: Sestav nenormovanou Laplaceovu matici L = D - W.
3: Sestav symetrickou normovanou Laplaceovu matici $L_{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$.
4: Vypočítej prvních k vlastních vektorů $u_1, ..., u_k$ z matice L_{sym}.
5: Nechť $U \in \mathbb{R}^{nxk}$ je matice obsahující vektory $u_1, ..., u_k$ jako sloupce.
6: Sestav matici $T \in \mathbb{R}^{nxk}$ z matice U normalizací řádků, tj.

$$T = (t_{ij})_{i,j=1}^n \quad \text{kde} \quad \frac{u_{ij}}{\left(\sum_k u_{jk}^2\right)^{\frac{1}{2}}}.$$

7: Pro i = 1,...,n nechť $y_i \in \mathbb{R}^k$ je vektor odpovídající i-tému řádku matice T.
8: Shlukni body $(y_i)_{i=1}^n$ do shluků $C_1, ..., C_k$ za použití k-means algoritmu.

3.3 Složitost

Sluková analýza může být použita pro širokou škálu aplikací a různě velké datové sady. Tyto data se mohou pohybovat od několika desítek až stovek datových bodů až po několik stovek tisíc až milionů. Z tohoto důvodu nás může zajímat informace o výpočetní složitosti algoritmů. Abychom toho dosáhli, podíváme se postupně na výpočetní složitost jednotlivých kroků spektrálních algoritmů. Pro začátek definujme n jako počet datových bodů a d jako jejich dimenzi.

Prvním krokem je konstrukce matice podobnosti. Za předpokladu, že počítáme každou podobnost s_{ij} mezi dvěma datovými body x_i a x_j nás bude výpočet stát pro tuto hodnotu $\mathcal{O}(d)$. Z tohoto důvodu bude sestavení matice podobnosti pro celý graf mít složitost $\mathcal{O}(n^2d)$.

Dalším krokem je výpočet vlastních vektorů. Tento problém je náročný a existuje více druhů algoritmů pro vyřešení tohoto problému. Běžný přístup jak řešit tento problém je pomocí Lanczosova algoritmu pro symetrické matice, nebo Arnoldiho algoritmem pro nesymetrické matice. Aniž bychom šli více do detailu, jak tento algoritmus funguje, můžeme odvodit složitost následovně:

$$\mathcal{O}(m^3) + (\mathcal{O}(nm) + \mathcal{O}(nt)) \cdot \mathcal{O}(p(mk)),$$

kde $m > k$ je nazýváno jako Arnoldiho délka, která je definována uživatelem a p je číslo restartů Anrolidho iterací. Jen pro představu o těchto hodnotách, m je obvykle několikrát větší než k. Pro více přesnějších informací o tom, jak může být tato složitost dokázána, se odkážeme na [4].

Posledním krokem je spuštění k-means pro shlukování vrcholů grafu v redukované reprezentaci. S každou iterací musíme spočítat vzdálenosti mezi každým bodem a každým centroidem shluku. To nám dává časovou složitost $\mathcal{O}(l \cdot nk^2)$, kde l je počet iterací pro k-means.

Celkově můžeme vidět, že sestavení matice podobnosti je nejdražší záležitostí a obvykle potřebujeme pracovat s velkými datasety. Existuje zde několik přístupů, jak se vypořádat s tímto problémem, například použití paralelních výpočetních metod [11] nebo spolehnutí se na aproximační algoritmy [12].

3.4 Použití

Shluková analýza se používá v různých situacích, zasahujících většinou do vědeckých oblastí jako je fyzika, biologie, medicína a je téměř nemožné, abychom vyjmenovali celý seznam použití. My se však chceme podívat podrobněji jen na některé z těchto příkladů.

V oblasti *segmentace obrazu* se obvykle snažíme rozdělit obraz na určité segmenty nebo detekovat hrany - užitečná metoda například v oblasti zpracování obrazu a v medicíně. Každý pixel z *n-x-m* pixelů obrázků reprezentuje datový bod, v našem případě vrchol grafu. Máme-li tedy obrázek o velikosti 100x100 pixelů, bude náš graf mít 10 000 vrcholů. Musíme se vypořádat s maticí podobnosti, jež reprezentuje náš obrázek jako graf, o velikosti 10000x10000.

V této diplomové práci budeme v případě obrázků konstruovat matice podobnosti obsahující jako hrany vzájemné podobnosti mezi každou dvojicí pixelů. Jiný přístup, jak počítat podobnosti mezi pixely, založeném na váženém PCA kernelu, můžeme vidět zde [32]. Jako váhy hran můžeme počítat z rozdílných vlastností, například hodnoty RGB nebo intenzitu pixelu [13]. V tomto případě se musíme vypořádat s velkými datovými kolekcemi, dokonce i s obrázky s nízkým rozlišením. Běžný způsob, jak vyřešit tento problém, je aplikovat shlukovací algoritmy jen na malé bloky obrázků a poté sloučit výsledné segmenty metodami jako jsou *stochastic ensemble consensus* - detailnější vysvětlení můžeme najít zde [14]. V této diplomové práci se budeme věnovat segmentaci obrazu pomocí spektrálního slukování v kapitolách 5.6 a 6.7.

Dalším možným použitím je v *teorii signálů*. Zvažme například audio nahrávku obsahující hlasy dvou lidí mluvících zároveň. V tomto případě chceme rozdělit tuto lineární směs signálů a odfiltrovat individuální hlasy. Tímto dostáváme velmi složitý problém zahrnující o mnohem více technik než je jen shluková analýza. Podrobnější rozbor tohoto problému můžeme nalézt v [15].

My se však v této práci budeme zabývat daty, které reprezentují neorientované, vážené grafy. Pod tímto typem dat si můžeme například představit graf sociální sítě, kde vrcholy grafu jsou uživatelé sociální sítě, a hrany značí jejich podobnost. Například počet poslaných zpráv, počet společných přátel, počet společných skupin atd. Další možnou alternativou mohou být studenti, kteří společně navštěvovali různá cvičení, či teroristé mající společné vazby. V kapitole 5.2 se budeme zabývat různými datovými kolekcemi, které budeme v této diplomové práci analyzovat.

4 Metody měřící kvalitu

V této kapitole probereme dva typy metod měřících kvalitu. Prvním typem budou metody, které měří kvalitu shlukování, tedy, jsou-li prvky zařazeny do správných shluků. Druhým typem jsou metody, které měří kvalitu rozdělení grafu po shlukování.

4.1 Metody měření kvality shlukování

Do této podkapitoly patří silhouette index a dunnův index. Oba indexy jsou založeny na výpočtu vzdáleností mezi vrcholy shluků.

4.1.1 Silhouette index

Tato metrika používá konceptu soudržnosti a oddělení pro hodnocení shluků. Používá při tom vzdálenost mezi uzly. Silhouette index pro daný vrchol i je dán následující rovnicí:

$$S(C_i) = \frac{\sum_{v \in C_i} S_v}{|C_i|}, \ kde \ S_v = \frac{b_v - a_v}{max(a_v, b_v)}, \tag{20}$$

kde a_v je průměrná vzdálenost mezi vrcholem v a všemi ostatními vrcholy ve stejném shluku jako je v, a b_v je průměrná vzdálenost mezi v a všemi vrcholy v nejbližších shlucích, ve kterých není v. Silhouette index pro daný shluk je průměrná hodnota siluet pro všechny začleněné vrcholy. Tento index může nabývat hodnot mezi -1 a 1 s tím, že čím vyšší hodnoty silhouette indexu dosáhneme, tím lépe.

Nicméně silhouette index představuje také určitá omezení. V první řadě je to velmi nákladná metrika na výpočty. Dalším omezením je její chování v případě výskytů tzv. singleton shluků. Jelikož singleton nemá žádné vnitřní hrany, jeho vnitřní vzdálenost bude 0, což způsobí nesprávné skóre silhouette indexu a to perfektní 1. Tato cesta shlukování s mnoha singletony může vést k vysokému skóre nehledě na kvalitu ostatních shluků [5].

4.1.2 Dunnův index

Dunnův index je další metodou pro hodnocení shlukování. Patří do stejné kategorie jako silhouette index. Jedná se o interní systém hodnocení, kde jsou výsledky založeny na samotných shluknutých datech.

Stejně jako u ostatních indexů z této kategorie je jeho cílem určit množinu shluků, které jsou kompaktní, s malými rozptyly mezi členy daného shluku a s propojením na ostatní shluky.

Pro dané shlukování, pokud je dunnův index vysoký, znamená to kompaktní a dobře rozdělené shluky. Nevýhodou je výpočetní náročnost pro velký počet shluků a rostoucí dimenze dat [8].

V této práci se budeme zabývat následujícím vzorcem pro výpočet dunnova indexu:

Nechť C_i je shluk dat. Nechť x a y jsou jakékoli dva n-dimenzionální vektory vlastností (featury vektory) [3] přiřazené do stejného shluku C_i.

Pro každý shluk vypočítáme vzdálenost mezi dvěma nejvzdálenějšími vrcholy uvnitř daného shluku (intra-cluster distance).

$$\Delta_i = \max_{x,y \in C_i} d(x, y).$$

Pro měření vzdáleností *d(x,y)* mohou být použity kterékoli z dobře známých metrik, jako například manhattanská vzdálenost, nebo euklidovská vzdálenost.

Podobně musíme provést měření s venkovními vzdálenostmi mezi shluky (inter-cluster distance). Použijeme tedy dva nejbližší vrcholy pro dané dva shluky. Nechť

$$\delta(C_i, C_j)$$

je inter-cluster vzdálenostní metrika mezi shluky C_i a C_j.

S výše uvedenou notací, pro m jako číslo počtu shluků, je dunnův index pro danou datovou sadu definován jako:

$$DI_m = \frac{\min\limits_{1 \leqslant i < j \leqslant m} \delta(C_i, C_j)}{\max\limits_{1 \leqslant k \leqslant m} \Delta_k}. \tag{21}$$

Avšak tato formulace vzorce pro dunnův index má typický problém s tím, že pokud jeden ze shluků má špatné vlastnosti, zatímco ostatní jsou „zhutněné“, bude mít dunnův index pro tento typ shluků netypicky nízkou hodnotu. Tohle je jakýsi indikátor nejhoršího případu a je třeba ho mít na paměti [8].

4.2 Metody měření kvality shlukování uzlů grafu

Problém může nastat s počátečním výběrem počtu shluků a měřením jejich kvality. Mohou nám pomoci hodnotící metriky jako je modularita (4.2.1), konduktance (4.2.2) a cut ratio (4.2.3). Každá z těchto metrik, na základě různých vlastností výpočtů, spadá do odlišných skupin definovaných v článku [7]. Článek [7] pojednává o tzv. „ground truth“ komunitách a popisuje různé způsoby měření dělení grafů, avšak žádná z těchto metod není nazvána jako nejlepší.

I přes snahu identifikovat kvalitní shluky, nejsou tyto metriky dokonalé. Je vlastně velmi obtížné určit, zda daná metrika dává odpovídající výsledky na dané shluky grafu, protože obvykle nemáme žádné výsledky k srovnání. To platí zejména pro větší datasety s reálnými daty [5].

V této části se seznámíme a porovnáme několik nejvíce známých metrik na měření kvality shlukování nad grafy. Bude vyhodnoceno, zda tyto metriky opravdu reprezentují klasický pohled na to co je shluk a jak se chovají ve větších, méně předvídatelných případech.

[3] N-dimenzionální vektor vlastností (featur), kde vlastností (featurou) můžeme rozumět nějakou individuální měřitelnou charakteristiku, a samotný vektor reprezentuje nějaký objekt. V případě obrázků mohou hodnoty featur znamenat hodnoty pixelů obrázků, v případě grafu například vzdálenosti mezi vrcholy grafu.

4.2.1 Modularita

Modularita je jedna z nejvíce populárních ověřovacích metrik pro shlukování v grafu. Udává kvalitu rozdělení komunit grafu. Její výpočet vychází z úvahy porovnání hustoty hran uvnitř jednotlivých shluků s celkovým počtem hran grafu [5].

Skóre modularity q pro shlukování je vyjádřeno následující rovnicí:

$$q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} deg(v)}{2m} \right)^2 \right], \tag{22}$$

kde $|E(C)|$ je součet vah vnitřních hran shluku[4], m je celková váha hran grafu a $\sum_{v \in C} deg(v)$ je suma stupňů vrcholů daného shluku[5] [9].

Index modularity q většinou prezentuje hodnotu mezi 0 a 1. S tím, že 1 reprezentuje shluk s velmi silnými vlastnostmi komunity. Nicméně, některé limitní případy mohou prezentovat záporné hodnoty. Jeden z případů je přítomnost shluku pouze s jedním vrcholem. V tomto případě tyto shluky mají 0 vnitřních hran, a proto zde není nic k měření. Dostatečně velký počet singleton shluků v daném shlukování může vést k naměření příliš nízké hodnoty, což zastíní ostatní, pravděpodobně lépe formované shluky z daného shlukování a může způsobit velmi nízkou hodnotu modularity [5].

4.2.2 Konduktance

Konduktance řezu je metrika, která porovnává velikost řezu (tj. počet hran řezu) a váhu hran v každém ze dvou podgrafů indukovaných tímto řezem. Konduktance $\phi(G)$ grafu je minimální konduktanční hodnota mezi všemi jeho shluky. Konduktance patří do kategorie metod, které kombinují interní a externí propojení [7].

Považujme řez, který rozdělí graf G do k nepřekrývajících se shluků $C_1, C_2, ..., C_k$. Konduktance každého shluku se spočte dle rovnice 23

$$f(S) = \frac{c_s}{2m_s + c_s}, \tag{23}$$

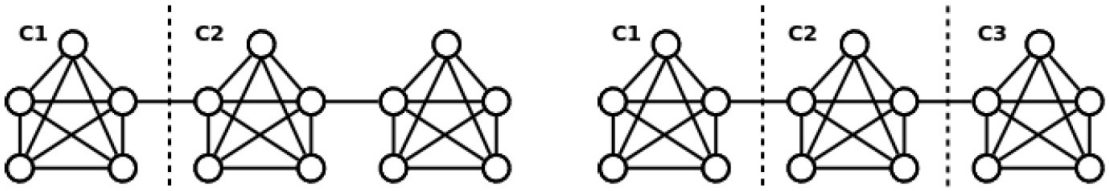
kde pro funkci f je S množina vrcholů daného shluku, c_s je součet vah hran vedoucích ze shluku, tedy hrany na hranici S a m_s je počet hran v podgrafu S [7].

V této práci budeme předpokládat konduktanci pro celé shlukování jakožto průměrnou hodnotu těchto řezů. V porovnání s modularitou nás bude zajímat co nejnižší konduktance.

Jedna negativní vlastnost konduktance poukazuje na to, že může mít tendenci dávat lepší skóre shlukováním s menším počtem shluků, i přes to, že více shluků bude mít pravděpodobně víc řezů hran. Nedostatek informací o interní hustotě hran používané v tomto typu konduktance

[4] Pouze ty hrany, které zasahují do vrcholů uvnitř shluku. Hrany, které sousedí s vrcholy v jiných shlucích se zde nezapočítávají.

[5] Pro neohodnocený graf je to počet hran, které do vrcholu vstupují. Pro ohodnocený graf suma vah hran, které do vrcholu vstupují.



Obrázek 6: Různá rozdělení grafu (na shluky uzlů) [5]

může vést k problémům, jak je znázorněno na obrázku 6, kde obě prezentovaná shlukování by měla stejná konduktanční skóre, dokonce i když je rozdělení vpravo z obrázku 6 zjevně lepší [5].

4.2.3 Cut Ratio

Posledním indexem, kterým se budeme zabývat v této diplomové práci, je cut ratio. Tento index spadá do kategorie založené na externím propojení. Cut ratio můžeme popsat jako zlomek existujících hran (ze všech možných hran) opouštějících shluk [7]. Je dán následující funkcí:

$$f(S) = \frac{c_s}{n_s(n - n_s)}, \tag{24}$$

kde, stejně jako v předchozí kapitole o konduktanci 4.2.2, pro funkci f je S množina vrcholů daného shluku, c_s je součet vah hran vedoucích ze shluku, tedy hrany na hranici S, n_s je počet vrcholů v podgrafu S a n je celkový počet vrcholů grafu [7].

Výsledná hodnota cut ratia pro celý graf bude v této práci průměrná hodnota ze všech vypočtených hodnot pro každý shluk. Čím nižší bude tato hodnota, tím lépe.

5 Vlastní implementace

V následující kapitole budou popsány desktopové aplikace určené pro práci s grafovými datovými kolekcemi a segmentaci obrazu pomocí spektrálních algoritmů. Dále představíme datový formát, ve kterém jsou uloženy grafy, které budeme analyzovat. Následně představíme knihovny třetích stran, které byly použity při implementaci aplikací pro tuto diplomovou práci. Dále bude zobrazen diagram komponent, ve kterém uvidíme jejich propojení a souvislosti. Poté budeme postupně procházet, na základě rozdělení projektu do komponent, implementací spektrálních shlukovacích algoritmů pro různá odvětví a následně metodami měřících kvalitu shlukování.

5.1 Popis desktopové aplikace

Pro implementaci byl zvolen .NET framework verze 4.5 společnosti Microsoft. S tímto frameworkem je spojeno vývojové prostředí Visual Studio, pro tuto diplomovou práci byla vybrána v počátku verze 2013, posléze 2015. Jako programovací jazyk bude použit jazyk C#. Aby byla aplikace použitelnější a lépe ovladatelná, budou naimplementována uživatelská rozhraní. Uživatelská rozhraní budou vytvořena pomocí technologie WPF, která slouží k tvorbě desktopových aplikací. Aplikace nám umožní jednodušeji volit parametry pro výpočty a lépe zobrazovat výsledky než jednoduchá konzolová aplikace.

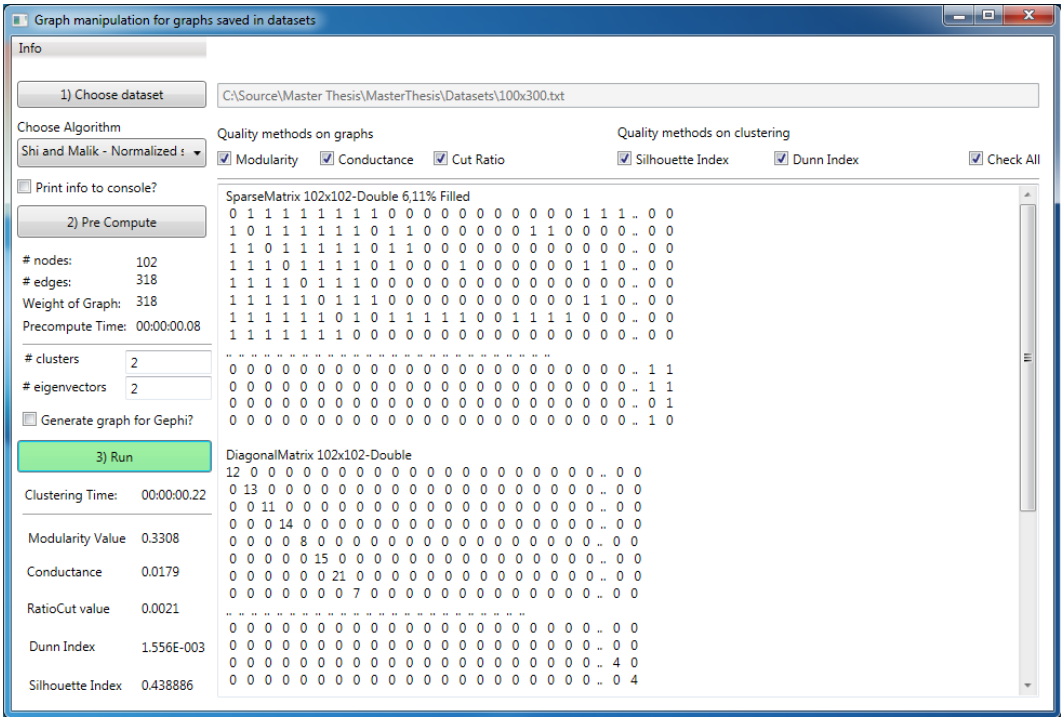
Uživatelské rozhraní se skládá ze dvou nezávislých oken. Jedno slouží pro práci s datovými kolekcemi obsahující grafy a druhé pro práci s obrázky. V obou oknech může uživatel volit různé parametry výpočtů.

5.1.1 Uživatelské rozhraní pro práci s grafy

Na obrázku 7 můžeme vidět podobu uživatelského rozhraní, jež je určeno pro práci s grafovými datovými kolekcemi, které budou popsány v následující kapitole 5.2.

Jednou z nejdůležitějších funkcí této aplikace je vybrání daného algoritmu, pomocí kterého chceme graf analyzovat. Jsou zde na výběr tři možnosti algoritmů, které byly popsány v kapitole 3. Další vlastností je výběr požadované datové kolekce. Jakmile je vybrána datová kolekce a algoritmus, vypočtou se požadované matice a vlastnosti dané datové kolekce. Poté se v levé části aplikace zobrazí, nad časem počáteční analýzy, informace o grafu - počet hran, počet vrcholů a celková váha hran grafu. Tyto matice a informace o grafu zůstanou uloženy v paměti a my máme možnost měnit pro vybraný algoritmus a dataset pouze počty shluků a počty vlastních vektorů. Změníme-li algoritmus, musí být znovu vypočteny všechny požadované matice a údaje o grafu.

Dále jsou zde dvě textová pole, jedno pro volbu počtu shluků a druhé pro volbu počtu vlastních vektorů pro matici obsahující vlastní vektory jako sloupce. Jak bylo zmíněno dříve, spektrální algoritmy jsou definovány tak, že pracují pouze s variantou, kde počet shluků je roven



Obrázek 7: Ukázka uživatelského rozhraní pro práci s grafy.

počtu vlastních vektorů. V kapitole o experimentech si uvedeme, proč jsme chtěli mít i volbu počtu vlastních vektorů.

Uživatel má možnost volby, které měřící metody chce použít. Tato zaškrtávací políčka jsou v horní části aplikace pod textovým polem, které zobrazuje vybranou datovou kolekci.

Ve spodní levé části jsou výpisy metod měřících kvalitu rozdělení grafu a shlukování. Na základě těchto výsledků bude v části o experimentech měněn počet shluků a počet vlastních vektorů s účelem najít nejlepší možný výsledek pro vybraný graf.

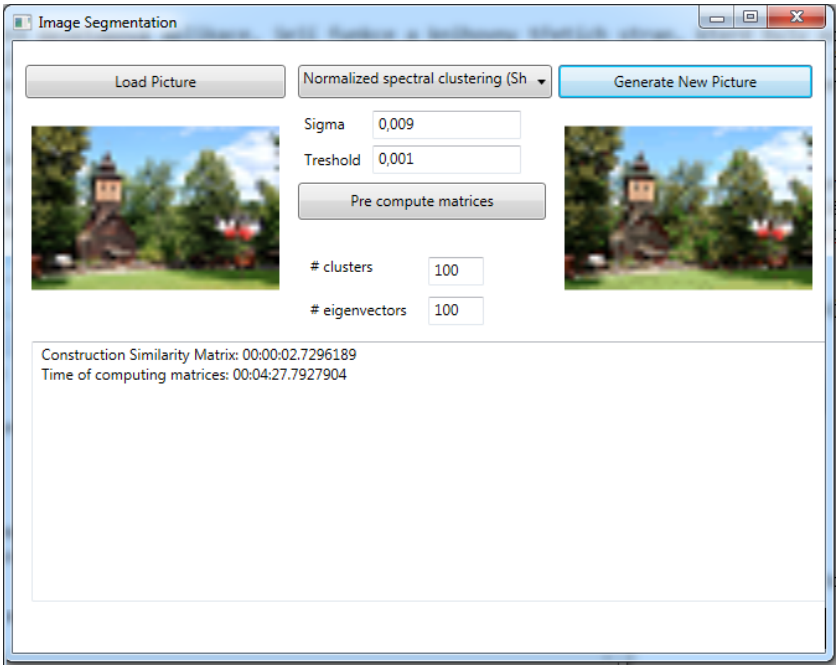
Uživatel má také možnost volby, zda chce vypisovat mezikroky výpočtů do textové oblasti, která zabírá většinu prostoru aplikace.

Zaškrtne-li uživatel tlačítko pro generování grafu, zobrazí se na konci výpočtu nové okno pro uložení vygenerovaného GraphML souboru pro vizualizační program Gephi, více v 6.1.2.

5.1.2 Uživatelské rozhraní pro práci s obrázky

Obrázek 8 znázorňuje uživatelské rozhraní, které slouží pro segmentaci obrazu pomocí spektrálních shlukovacích algoritmů. Základní funkcionalita umožňuje načíst požadovaný obrázek (levá část okna), poté vybrat algoritmus a na základě zvolených parametrů *sigma* a *treshold* zkonstruovat matici podobnosti (prostřední část okna). Pro urychlení výpočtů experimentů je zde tlačítko, které na základě vybraného algoritmu a zadaných parametrů *sigma* a *treshold* sestrojí požadované matice a ponechá je uložené v paměti. Poté můžeme experimentovat s počty k shluků a l vlastních vektorů nad stejnou maticí podobnosti a laplaceovou maticí. Tlačítko v pravé části

okna vygeneruje nový obrázek. Spodní část okna slouží k vypisování informací o časech výpočtů matic a shlukování.



Obrázek 8: Ukázka uživatelského rozhraní pro práci s obrázky.

5.2 Datové kolekce a jejich formát

Datové kolekce, které budeme analyzovat, obsahují neorientované grafy. Jedna část těchto grafů jsou vážené, mají tedy u každé hrany uvedenou váhu mezi dvěma vrcholy, a nebo neválené grafy, jejímž hranám budeme přiřazovat váhu s hodnotou 1.

Datové kolekce je možnost nalézt například na stránkách univerzit, které poskytly tato data k testovacím účelům. Máme možnost stáhnout data z university *Stanford*, která obsahuje různě rozsáhlé datové kolekce v různých formátech. Tato data mají na starost pánové Jure Leskovec a Andrej Krevl [24]. V této diplomové práci však budou použita data z německé university *Koblenz* [25].

Data z Koblenzské university reprezentují různé problémy (oblasti). Budeme analyzovat datovou sadu obsahující kontakty mezi podezřelými teroristy zapojených do bombových útoků na vlak v Madridu v březnu roku 2004. Vrchol zobrazuje teroristu a hrana informaci, zda byl mezi nimi kontakt. Váha hran znázorňuje, jak silná vazba mezi nimi byla. Zahrnuje přátelství a spolupráci v tréninkových kempech nebo předchozích společných útocích.

Další datové kolekce, které budou analyzovány, jsou různě rozsáhlé syntetické sítě, které byly generovány programem modelující sítě s požadovanými vlastnostmi, které jsou specifikovány pomocí vstupních parametrů. Tyto sítě poskytl pan doc. Mgr. Miloš Kudělka, Ph.D.

Všechna tato data jsou v podobném formátu, který je zobrazen v následující ukázce:

```
0 1 8
0 2 3
0 5 1
1 2 4
```

kde na každém řádku je záznam pro jednotlivou hranu. V prvním a druhém sloupci jsou názvy uzlů a ve třetím sloupci jsou váhy hran. Pokud nejsou v datové kolekci v třetím sloupci uvedeny váhy hran, bude váha hrany 1.

Dalšími datovými kolekcemi budou obrázky, kde je každý pixel považován za vrchol grafu a hrany mezi pixely značí jejich podobnost [13].

5.3 Použité knihovny třetích stran

V počátečních fázích implementace bylo k problémům, jež jsou vázány k práci s grafy, přistupováno poměrně naivně a bez jakéhokoli důrazu kladeného na omezení paměti počítače. Při ukládání vážené matice sousednosti byl brzy objeven problém oznámen výjimkou *System.OutOfMemoryException* s nedostatkem paměti při ukládání velkých grafů. Museli jsme proto přejít k ukládání grafů do řídké matice. S tímto problémem nám pomohla knihovna *Math.NET Numerics*, které se věnuje následující kapitola 5.3.1.

Tato knihovna byla však opět omezena svým výkonem při počítání vlastních čísel a vlastních vektorů a proto byla použita knihovna *Alglib*, které se věnuje kapitola 5.3.2.

5.3.1 Matematická knihovna pro Math.NET Numerics

Tato knihovna, jak již její název napovídá, je určena pro .NET framework. Lze ji použít ve dvou základních balíčcích:

- MathNet.Numerics - hlavní balík,
- MathNet.Numerics.FSharp - optimální rozšíření pro lepší práci při použití jazyka F#.

Tato knihovna byla zvolena především kvůli jednoduché manipulaci a lehkému ovládání. Prvním důvodem použití byla možnost pracovat s různými maticemi. V první řadě s řídkými maticemi, do kterých byla potřeba ukládat velké grafy, přesněji tedy matice zahrnující informace o grafech. Tyto matice, u velkých grafů obsahující miliony prvků, z velké části obsahují nuly a řídké matice nám v tomto případě pomohou tak velkou strukturu uložit lépe do paměti, aniž by nastala chyba s nedostatkem paměti počítače.

Řídké matice ukládají nenulové elementy ve třech polích na základě standardního komprimovaného řídkého formátu **CSR**. Jedno pole ukládá všechny hodnoty, které jsou nenulové, další pole stejné délky ukládá jejich odpovídající index sloupce. Třetí pole, jehož délka je počet řádků + 1, ukládá informace o tom, kde začíná každý řádek. Celkový počet nenulových prvků je v posledním prvku pole [20].

Tyto řídké matice budou použity při konstrukci matic W i Laplaceových matic, které z podstatné části také obsahují nuly. Diagonální matice D bude ukládána do dalšího speciálního formátu, více v 5.5. Problém může nastat u matic, které však neobsahují téměř žádné nuly. Užití řídké varianty zpomaluje výpočet a je na zváženou, zda u některých matic nepoužít hustou strukturu namísto řídké [20].

Nedostatečně efektivní řešení bylo shledáno při použití výpočtu vlastních čísel a vlastních vektorů pomoci této knihovny. Výpočty fungovaly správně a knihovna vracela výsledky, které měly být vráceny [6]. Problém byl shledán v časech výpočtů. Tato knihovna prováděla výpočty řádově v minutách místo v sekundách. Tato skutečnost je ovšem velmi neefektivní a u větších grafů se několikanásobně zpomaloval výpočet. Museli jsme tedy přejít k jinému řešení. Jako první se naskytla velmi robustní, ověřená a používaná knihovna jménem **Alglib**.

5.3.2 Matematická knihovna Alglib

Alglib je multiplatformní knihovna pro číselnou analýzu a zpracování dat. Podporuje několik programovacích jazyků (C#, C++, Pascal, VBA) a několik operačních systémů (Windows, Linux, Solaris). Alglib poskytuje tyto možnosti:

1. analýzu dat (klasifikace/regrese, neuronové sítě),
2. optimalizace a nelineární řešitelé,
3. interpolace a lineární / nelineární metody nejmenších čtverců,
4. řešení problémů z lineární algebry (EVD/SVD), přímé a iterativní lineární řešitelé a mnoho dalších, speciálních funkcí a algoritmů.

Alglib je možno použít ve dvou edicích. První edice je zdarma, avšak neumožňuje tak výkonné výpočty jako komerční verze. Komerční verze umožňuje více vláknové výpočty a 100% řízený kód. Komerční verze navíc poskytuje vysoce výkonnou verzi pro jazyk C++ [21]. Pro tuto diplomovou práci bude použita free verze 3.10.0 pod licencí GPL v jazyce C#.

Tato knihovna byla použita v prvním případě pro výpočet vlastních čísel a vlastních vektorů Laplaceových matic. Alglib poskytuje více metod pro tento výpočet. V této diplomové práci byly použity dvě metody. Jedna metoda pracuje se symetrickou maticí a druhá s nesymetrickou. Pro tuto diplomovou práci byly použity obě metody, protože jak bylo zmíněno dříve, dvě Laplaceovy matice jsou symetrické (L a L_{sym}) a jedna nesymetrická (L_{rw}). Metoda pro výpočet vlastních čísel a vlastních vektorů ze symetrické matice používá *QL/QR algoritmus*. Druhá metoda *QR algoritmus s vícenásobnými posuny*. Zatímco symetrická varianta vrací reálná vlastní čísla a matici obsahující všechny vlastní vektory, nesymetrická varianta vypočítává reálnou i imaginární část a dvě matice obsahující levé i pravé vlastní vektory. V této diplomové práci je počítáno pouze s pravými vlastními vektory [22].

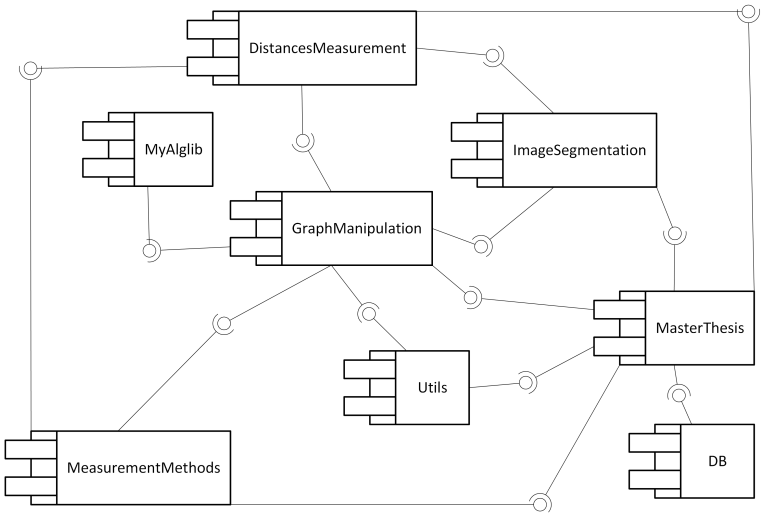
[6] Kontrola výsledků probíhala pomocí statistického programu R. Více o těchto kontrolách v kapitole 6.2.

Alglib nám dále pomohl se samotným shlukováním. Alglib v této problematice nabízí dvě možnosti, jak datové prvky rozdělit do shluků. Prvním je *aglomerativní hierarchické shlukování* a druhým *k-means*. Pro tuto diplomovou práci byla zvolena druhá varianta. Teorie o tomto shlukovacím algoritmu byla popsána v kapitole 2.6.1. Alglib používá jeho vylepšenou variantu k-means++, která je přesnější a rychlejší než k-means [34].

Jak byly tyto metody použity a podrobnosti, jak o výpočtech vlastních problémů, tak shlukování, najdeme v následujících kapitolách o implementaci spektrálních algoritmů.

5.4 Diagram komponent

Diagram komponent na obrázku 9 znázorňuje rozdělení aplikace do jednotlivých komponent, do kterých byla aplikace pro přehlednost rozdělena. Jsou zde zobrazeny jednotlivé závislosti.



Obrázek 9: Diagram komponent.

5.5 Komponenta GraphManipulation

Prvním krokem, který bylo potřeba udělat při práci s grafovými datovými kolekcemi, bylo převést vstupní soubor s grafem do jiné podoby. K tomuto účelu slouží třída *GraphParser*. Při analýze souboru jsou jednotlivé vrcholy ukládány do objektů třídy *Node* a všechny hrany do objektů *Edge*. Pokud *parser* nalezl vrchol, který již měl uložen, znovu jej neukládal pokračoval na další vrchol. Poté byla hrana uložena k oběma vrcholům. Struktura byla navržena tak, že ke každému vrcholu je ukládán seznam jeho hran do kolekce *List<Edge>*. Informace o všech těchto vrcholech a hranách jsou ukládány ve třídě *Graph*.

Jakmile byl soubor grafu úspěšně analyzován, mohlo se přejít ke konstrukci matic W a D. Funkce pro jejich konstrukci jsou součástí třídy *Graph*. Ke konstrukci řídké matice sousednosti slouží metoda *public Matrix<double> CreateSparseWeightedAdjancedMatrix()*, která na základě sousedících vrcholů a jejich vah hran matici sestaví.

Dalším bodem, který bylo potřeba vyřešit, je ukládání diagonální matice *D*. V tomto případě jsou hodnoty stupňů vrcholů uloženy pouze na hlavní diagonále matice a zbytek elementů jsou nuly. Knihovna Math.NET Numerics poskytuje pro tento problém třídu *DiagonalMatrix*. Hodnoty na diagonále jsou ukládány v jednorozměrném poli. V našem případě stačilo konstruktoru předat počet řádků, počet sloupců a pole prvků, jež ukládá stupeň vrcholu každého vrcholu grafu. Konstrukci diagonální matice *D* má na starost metoda *public DiagonalMatrix CreateSparseDiagonalMatrix()* třídy *Graph*.

5.6 Implementace segmentace obrazu a komponenta ImageSegmentation

V případě datové kolekce obsahující pixely, musíme konstruovat matici sousednosti *W* jiným způsobem, než v případě datasetů obsahující grafy. U segmentace obrazu pomocí spektrálního shlukování je každý pixel reprezentován jako vrchol grafu a hrany mezi vrcholy obsahují hodnotu jejich podobnosti. Čím vyšší hodnota, tím více jsou si pixely podobné [13]. V tomto případě musíme počítat podobnost mezi jednotlivými pixely a tuto podobnost ukládat jako váhu hrany. Pro práci s pixely slouží komponenta *ImageSegmentation*, avšak informace o pixelech jsou ukládány do objektů třídy *Node*. Jsou zde pro ně odlišné vlastnosti a to pozice X a pozice Y znázorňující jejich pozici na originálním obrázku a poté jejich originální barva a barva v odstínu šedi, do kterého byla původní barva pixelu převedena.

Pro manipulaci s obrázky slouží třída *ImageManipulation*. Tato třída obsahuje metodu *public void AllocatePixels(Bitmap bmap)*, která v počáteční inicializaci prochází všechny pixely obrázku a ukládá o nich dříve zmíněné vlastnosti. Další metody této třídy převádí *BitMapImage* na *BitMapu* a naopak pro zobrazení v uživatelském rozhraní či pro uložení obrázku po dokončení algoritmu.

Jakmile máme uloženy informace o všech pixelech v objektech třídy *Node*, začneme konstruovat matici sousednosti *W* výpočtem podobnosti pomocí Gaussovy podobnostní funkce (rovnice 3), známé také jako gaussovský kernel. K tomuto účelu slouží metoda *public Matrix<double> CreateSparseSimilarityMatrix(double sigma, double treshold)*, která je součástí třídy *Graph*. Tato metoda prochází všechny dvojice pixelů obrázku a na základě hodnot R,G,B a A původního pixelu počítá jejich vzdálenost pomocí metody *public static float Calculate(int[] vect1, int[] vect2)*, která je součástí třídy *EuclideanDistance* komponenty *DistanceMeasurement*. Máme-li vypočtenou tuto vzdálenost, bude váha hrany vypočtena následovně: *var edgeWeight = Math.Exp(-sigma * distance);*. Kde parametr sigma je vstupním parametrem funkce konstruující matici podobnosti.

Abychom mohli vytvořit řídkou matici, zanedbáváme podobnosti nižší než je parametr *treshold*, který je také vstupním parametrem funkce.

Pro vytvoření diagonální matice D slouží metoda *public DiagonalMatrix CreateDiagonalMatrixFromSimMatrix(Matrix<double> similarityMatrix)*, která je také součástí třídy *Graph*.

5.7 Implementace spektrálních algoritmů

Problematika spektrálních rozkladů Laplaceových matic a jejich implementace je společná pro segmentaci obrazu i grafové datasety. Obě problematiky využívají stejných tříd a metod. Jedná se především o třídu *SpectralClustering* v komponentě *GraphManipulation*.

5.7.1 Nenormovaný spektrální algoritmus

Jak získat matici L zobrazuje rovnice 4. Laplaceovu matici ukládáme, stejně jako matici W, do řídké struktury. Knihovna *Math.NET Numerics* poskytuje základní operace s maticemi a proto vznikla metoda *public static Matrix<double> CrateLaplacianUnnormalizedMatrix(Matrix <double> WeightedAdjancedMatrix, DiagonalMatrix DiagonalMatrix)* ve třídě *SpectralClustering*, která matice odečte a vrátí novou Laplaceovu matici.

Jakmile získáme tuto matici, další krokem algoritmu je zkonstruování nové matice, matice U. Tato matice obsahuje jako sloupce vlastní vektory příslušné vlastním číslům Laplaceovy matice.

Metoda počítající pomocí knihovny Alglib vlastní čísla a vlastní vektory ze symetrické matice má název *public static void SetMatrixOfEigenVectorsFromSymMatrix(double[,] symetricMatrix, int n, Graph graph)* a její hlavní část vypadá následovně:

```
alglib.smatrixevd(symetricMatrix, n, 1, isupper, out d, out z);
```

V této ukázce můžeme vidět, jaké parametry metoda *smatrixevd* přijímá. Jedná se o symetrickou matici, počet řádků matice, zda má algoritmus vypočítat vlastní vektory, úložný formát, výstupní pole d s uloženými vlastními čísly, a výstupní parametr z, kde jsou uloženy vlastní vektory jako sloupce.

Původní algoritmus pracuje s variantou, že počet shluků je roven počtu vlastních vektorů. My se v kapitole o experimentech zaměříme na tu možnost, že počet vlastních vektorů se bude lišit od počtu shluků a na základě měřících metod vyhodnocovat výsledná rozdělení vrcholů grafů do shluků.

Proto metoda *public static double[,] GetMatriU(double[,] matrixOfEigenVectors, int n, int kEigenVectors)* konstruující matici U má tolik sloupců, kolik vloží uživatel jako vstupní parametr pro počet vlastních vektorů, nikoli počet shluků. Metoda navíc ignoruje jednotkový vektor příslušný nejmenšímu vlastnímu číslu a bereme v potaz vlastní vektory začínající tzv. *Fiedlerovým* vektorem, jež je vektor příslušný druhému nejmenšímu vlastnímu číslu.

5.7.2 Normovaný spektrální algoritmus Shi a Malika

Tento algoritmus potřebuje pro svůj výpočet Laplaceovu matici L. O konstrukci této matice pojednávala předchozí kapitola, proto tento krok bude vynechán. Dalším krokem je konstrukce Laplaceovy matice L_{rw}. Tuto matici, jak uvádí kapitola 2.4.2, dostaneme z inverzní diagonální matice D vynásobenou s maticí L. Se získáním inverzní matice nám opět pomohla knihovna

Math.NET Numerics, která poskytuje řadu metod pro práci s maticemi, mimo jiné i metodu *Inverse()*. Následujícím krokem byla konstrukce matice U, obsahující vlastní vektory jako sloupce. Rozdíl v této metodě, oproti získání vlastních vektorů z Laplaceovy matice L, je v tom, že matice L_{rw} není symetrická. Proto musela být použita jiná metoda knihovny Alglib než je zmiňována v předchozí kapitole. Metoda pro výpočet vlastních čísel a vlastních vektorů knihovny Alglib se jmenuje *alglib.rmatrixevd(nonsymetricMatrix, n, vneeded, out wr, out wi, out vl, out vr)*. Tento kód je vyňat z metody *public static void SetMatrixOfEigenVectorsFromNonSymMatrix(double[,] nonsymetricMatrix, int n, Graph graph)*, která je součástí třídy *SpectralClustring* a vypočte vlastní čísla a vlastní vektory z nesymetrické matice. Jak je vidět, tato metoda přijímá více parametrů než metoda z předchozí kapitoly, jež počítala vlastní vektory ze symetrické matice. Je to proto, že v tomto případě vlastní čísla mohou, krom hodnot reálných, nabývat i hodnot imaginárních. Dalším rozdílem je to, že jsou zde počítaný levé vlastní vektory i pravé vlastní vektory. Jak již bylo zmíněno dříve, v této diplomové práci se berou v potaz pouze pravé vlastní vektory. Jak uvidíme v kapitole o experimentech, výpočet vlastních vektorů z nesymetrické matice je časově náročnější než výpočet z matice symetrické.

Metoda *GetMatrixU* konstruující matici U obsahující vlastní vektory jako sloupce vypočtených z matice L_{rw} je stejná jako v předchozí kapitole.

5.7.3 Normovaný spektrální algoritmus Ng, Jordana a Weisse

Tento algoritmus vychází z odborného článku [2] a je popsán v kapitole 3.2. Pro tento algoritmus potřebujeme opět zkonstruovat novou Laplaceovu matici vycházející z matice L a to matici L_{sym}. Následující kód znázorňuje tvorbu dvou matic. První maticí je matice $D^{-\frac{1}{2}}$ sestavena z diagonální matice D a druhá matice je L_{sym}.

```
var D = graph.DiagonalMatrix.PointwisePower(0.5);
graph.Lsym = D * graph.UnnormalizedLaplacianMatrix * D;
```

Výpis 1: Ukázka konstrukce dvou matic pomocí knihovny Math.NET Numerics

V ukázce vidíme použití metody *PointwisePower*, jež přijímá hodnotu záporného exponentu a opět spadá do kategorie metod knihovny Math.NET Numerics, které operují s maticemi.

Poté následuje stejný krok jako u předchozích matic a to výpočet vlastních čísel a vlastních vektorů uložených jako sloupce v nové matici U vypočtených z matice L_{sym}. Na rozdíl od předchozím dvou algoritmů je zde ještě jeden krok navíc. Musíme zkonstruovat novou matici T, jejíž dimenze je stejná jako matice U, avšak její řádky projdou normalizací. Vzorec pro onu normalizaci je uveden v kapitole 3.2.3 v 6. kroku algoritmu a vychází z článku [2]. Metoda konstruující novou matici T se jmenuje *public static double[,] GetMatrixT(double[,] matrixU, int n, int kEigenVectors)* a na základě vložené matice U sestrojí novou matici. Tato metoda je opět součástí třídy *SpectralClustering*.

5.7.4 Shlukování pomocí k-means++

Získáme-li matici U, případně matici T, můžeme překročit k dalšímu kroku algoritmu a to shlukování za použití algoritmu *k-means++* knihovny Alglib. Pro shlukování slouží metoda v následující ukázce. Tato metoda se používá pro všechny tři spektrální algoritmy.

```
public alglib.kmeansreport ClusteringReport(double[,] MatrixU, int kClusters)
{
      alglib.clusterizerstate state;
      alglib.kmeansreport report;
      alglib.clusterizercreate(out state);
      alglib.clusterizersetpoints(state, MatrixU, 2);
      alglib.clusterizersetkmeanslimits(state, 15, 0);
      alglib.clusterizerrunkmeans(state, kClusters, out report);
      return report;
}
```

Výpis 2: Ukázka metody přiřazení vrcholů do shluků pomocí algoritmu k-means++ na základě vložené matice U nebo T

Tato metoda je opět součástí třídy *SpectalClustering* a vrací report, který obsahuje důležité informace o shlucích. Abychom tyto informace dostali a mohli s nimi pracovat, potřebujeme shlukovacímu algoritmu říct, kolik shluků má vytvořit a jaká data shlukovat. Volí se zde počet restartů algoritmu a vzdálenostní metrika - v tomto algoritmu je na výběr pouze euklidovská vzdálenost. V reportu jsou uloženy centroidy shluků v matici kxn, kde k je počet shluků a n je počet sloupců matice U, na základě které se vrcholy přiřazují do shluků. Nejdůležitějším výstupním prvkem, se kterým budeme nadále pracovat, je jednorozměrné pole obsahující čísla shluků, do kterých jsou jednotlivé vrcholy zařazeny.

5.7.5 Barvení vrcholů grafu po shlukování

Barvení vrcholů grafu řešíme odlišným způsobem pro grafové datasety a pro obrázky, u obou však vycházíme z k-means reportu a jeho pole obsahující čísla shluků pro vrcholy grafu.

V případě obrázků procházíme postupně všechny shluky a vybereme původní barvu prvního pixelu, který se ve shluku nachází a uložíme ji do listu barev *List<Color> colors*. Máme-li dva shluky, pracujeme pouze s bílou a černou barvou. Poté procházíme postupně všechny pixely a dáváme jim novou barvu příslušnou jeho shluku. Poté tento obrázek převedeme na Bitmapu a vizualizujeme v uživatelském rozhraní. Tento problém řeší metoda *public BitmapImage GetClusteredImage(int width, int height, alglib.kmeansreport report, Graph graph, int kClusters)*, která je součástí komponenty *ImageSegmentation* a její třídy *ImageManipulation*.

Barvení grafů řešíme odlišným způsobem. V komponentě *Utils* máme metodu *public static List<string> GetColors(bool knownColors)*, která pracuje s dvěma variantami. Pokud chceme

obarvit graf méně než 30 shluky, máme zde 30 vygenerovaných barev. Chceme-li, aby měl graf více než 30 shluků, necháme si vygenerovat všechny známé barvy knihovnou *System.Drawing* a její třídou *KnownColor*. Tato generace potřebných barev je v metodě *public static XDocument GenerateGraphMlFile(string GraphMLFile, Graph graph, int[] clustersNumbers, int kClusters)*, která zkonstruuje XML soubor ve formátu *GraphML* pro vizualizační program *Gephi*. Popisu vizualizačního nástroje *Gephi* se věnuje kapitola 6.1.2.

5.8 Implementace metod měření kvality

Jak již bylo zmíněno dříve, tyto metody se dělí do dvou kategorií. Abychom dosáhli požadovaných výsledků, byla naprogramována pomocná třída *ClusterData* v komponentě *GraphManipulation*. Tato třída ukládá všechny potřebné informace o shlucích, které budou potřeba pro měřící metody. Společné vlastnosti napříč všemi metodami jsou *ID* shluku a všechny vrcholy, které patří do daného shluku. Další metody a vlastnosti jsou specifické pro různé metody měření a budou zmíněny v následujících podkapitolách. Měřící metody jsou součástí komponenty *MeasurementMethods*.

5.8.1 Modularita

Pro tuto metodu bylo potřeba ukládat následující vlastnosti: *součet vah vnitřních hran shluku* a *sumu stupňů vrcholů ve shluku*. V této třídě s názvem *Modularity* probíhal výpočet všech potřebných vlastností napříč všemi shluky. Metoda pro výpočet má dva parametry. Prvním je *List* všech shluků a druhým je *celková váha hran grafu*. Tyto vlastnosti musí být vypočítány pro každý shluk před výpočtem samotné modularity. Pro výpočet součtu vah hran byla ve třídě *ClusterData* naimplementována metoda *ComputeSumOfIntraClusterEdgesWeights()*, která procházela všechny vrcholy a hrany shluku a počítala váhy jen těch hran, které byly propojením jen těch vrcholů, které byly uvnitř shluku. Následná suma musela být vydělena dvěma kvůli výpočtu všech vah hran dvakrát. V počáteční fázi se modularita zdála náročná na naprogramování a výsledky neodpovídaly realitě. Problém byl v nedostatečně srozumitelném popsání elementů vzorce a následném pochopení. V počáteční fázi funkce počítala i ty hrany, které směřovaly ven ze shluku. V tomto případě hodnota modularity narůstala s větším počtem shluků místo toho, aby klesala při špatném rozdělení vrcholů grafu do shluků.

5.8.2 Konduktance a Cut Ratio

Pro práci s těmito metodami je naimplementována funkce *ComputeCountOfEdgesFacingOutOfCluster()*, která byla téměř totožná s funkcí pro výpočet vlastností modularity. Tato metoda však počítala hrany, které jsou spojené jak s vrcholem uvnitř shluku, tak s vrcholem vně shluku. Metoda pro výpočet konduktance, která je ve třídě *Conductance*, přijímá pouze *List* všech shluků a jsou zde potřeba následující vlastnosti: *počet hran uvnitř shluku* a *počet hran vedou-*

cích ze shluku. Na základě těchto vlastností stačilo dosadit do vzorce uvedeného v kapitole 4.2.2 a ukládat výsledky do *Listu doublů* a poté ze všech těchto hodnot vypočítat průměrnou hodnotu.

Metoda měřící cut ratio je ve třídě *CutRatio* a jako parametry také přijímá *List* všech shluků a navíc, oproti konduktanci, *počet vrcholů grafu*. Na základě těchto hodnot ukládáme do *Listu doublů* hodnoty pro každý shluk a poté všechny hodnoty o všech shlucích také zprůměrujeme.

5.8.3 Silhouette index

Pro výpočet silhouette indexu slouží třída *SilhouetteIndex* komponenty *MeasurementMethods*. Metoda *public static double GetSilhouetteIndexValue(List<Node> graphAllNodes)* počítá průměrnou hodnotu siluet všech vrcholů grafu. Pro každý vrchol grafu používáme ve třídě *Node* dvě vlastnosti. Jsou to *DistanceToClosestCluster* a *AverageDistanceBetweenOtherPointsIncluster*. První vlastnost pro každý vrchol určuje, na základě výpočtu vzdáleností, *nejbližší sousední shluk*, který značí další nejvhodnější shluk, do kterého by teoreticky mohl vrchol patřit. Výpočet této vlastnosti má na starost metoda *public void SetAvarageDistansesForEachNodeInclusterToDiffClusters(List<ClusterData> otherClusters, double[,] MatrixU, List<Node> allNodes)*, která je součástí třídy *ClusterData*. Druhá vlastnost znázorňuje průměrnou vzdálenost mezi ostatními vrcholy shluku, do kterého vrchol patří a počítá ji metoda *public void SetAverageClusterDistanceForEachNode(double[,] MatrixU)* třídy *ClusterData*. Vzdálenost mezi vrcholy jsou počítány euklidovskou vzdáleností na základě matice U, tedy matice obsahující l vlastních vektorů.

5.8.4 Dunnův index

Pro výpočet dunnova indexu slouží metoda *public static double GetDunnIndexValue(List <ClusterData> clusters, double[,] matrixU)* třídy *DunnIndex* komponenty *MeasuementMethods*. Na základě rovnice z kapitoly 4.1.2 potřebujeme pro každý shluk dvě vlastnosti: První *maxIntraClusterdist*, značící maximální vzdálenost mezi dvěma shluky, kterou počítá metoda *public void GetMaxIntraClusterDistanceFromMatrixU(double[,] MatrixU)* třídy *ClusterData*. Druhou *minInterClusterDistance*, značící minimální vzdálenost mezi jednotlivými shluky. Vzdálenost, která je počítána pro každou dvojici shluků, má na starost třída *SingleLinkage*. Single linkage je jednou z metod, jak vypočítat vzdálenosti mezi shluky a je spojována s hierarchickým shlukováním [7]. Metoda počítající single linkage nám tedy vrátí pro každou dvojici shluků vzdálenost mezi dvěma nejbližšími vrcholy z těchto dvou nepřekrývajících se shluků.

[7] Dalšími metodami jsou complete linkage a average linkage.

6 Experimenty a vizualizace výsledků

Kapitola se bude zabývat experimenty jak algoritmů využívajících spektrálních rozkladů, tak algoritmů měřících kvalitu. Budou představeny nástroje pro vizualizaci datových kolekcí, které byly uvedeny v kapitole 5.2. Převážná část kapitoly bude zaměřena na analýzu těchto datasetů. Bude zmíněn statistický programovací jazyk R, který pomohl v počáteční fázi experimentů s kontrolou výsledků. Poslední část této kapitoly se bude zabývat experimenty na barevných obrázcích.

6.1 Vizualizační nástroje

V následujících podkapitolách budou představeny dva open-source softwary pro vizualizaci grafů.

6.1.1 GraphViz

GraphViz je volně dostupný open-source software pro vizualizaci grafů pod licencí Eclipse Public License. Grafy jsou zobrazovány pomocí DOT formátu. Software umožňuje ukládat grafy do formátů PNG, SVG, PDF, PostScript a dalších. GraphViz umoňuje vybrat z několika layoutů[8]. Základním nástrojem pro vizualizaci je program *dot* ovládaný z příkazové řádky. Mezi další layouty, pomocí kterých se dá graf vykreslit, patří *neato*, který je lepší pro vykreslování grafů než dříve zmiňovaný *dot*. Pro velké neorientované grafy se doporučuje použít layout *sfdp* [33].

Pro jednodušší práci s tímto software byl použit GraphViz C# Wrapper[18]. I přes tato zjednodušení nesplňoval GrahpViz naše požadavky a práce s ním nebyla taková, jak jsme si v počáteční fázi představovali. Velké datové kolekce byly po vizualizaci nepřehledné a nedalo se s nimi pracovat.

Software Graphviz se hodí více pro práci a vizualizaci konečných automatů, případně UML diagramů.

6.1.2 Gephi

Po neúspěšném experimentu s vizualizací grafů pomocí knihovny GraphViz jsme přešli k robustnější, více propracovanější variantě vizualizace, open-source řešení Gephi, který je naprogramován v jazyce Java. Potřebuje tedy Javu ve verzi minimálně 7 a vyšší. V počátečních fázích psaní této diplomové práce jsme pracovali s verzí 0.8.2beta, která vyšla na začátku ledna 2013. V průběhu práce, v prosinci 2015, vyšla nová verze 0.9.0 a přešli jsme tedy k této verzi.

Gephi podporuje několik grafových formátů, pomocí kterých můžeme naše data tomuto programu předat. Formáty a jejich možnosti můžeme nalézt na [16].

Pro tuto diplomovou práci byl vybrán formát GraphML, protože splňuje všechny naše požadavky, především tedy XML strukturu, váhu hran a vizualizační atributy. Více informací o formátu GraphML nalezneme zde [19].

[8] Layoutem chápeme algoritmus pro vykreslení grafu.

Gephi obsahuje několik vizualizačních layoutů. Každý z layoutů se hodí pro vizualizaci jiného problému. Pro naše grafy se osvědčil Force Atlas a Force Atlas 2.

Každý z layoutů obsahuje parametry, které je potřeba nastavit pro co nejlepší vizualizaci. Tyto parametry ovlivňují například vzdálenost mezi vrcholy, sílu odporu, setrvačnost. V případě layoutu Force Atlas bylo manipulováno především s parametrem síly odporu, který vzdaluje na základě uvedené hodnoty vrcholy od sebe, kdy při nízké vložené hodnotě jsou vrcholy velmi blízko sebe a překrývají se, dokonce i s vrcholy z jiných shluků. Vložená hodnota závisí na velikosti grafu. Se zvětšující se hodnotou můžeme lépe vidět, jak jsou shluky od sebe vzdáleny a který vrchol patří do kterého shluku.

Kdybychom však chtěli zobrazit vrcholy jako body na mapě, lze stáhnout layout s názvem Geo Layout, kterému můžeme specifikovat zeměpisnou šířku a délku.

Tento open-source program navíc obsahuje i funkce, které vypočítají například modularitu grafu nebo jeho shlukovací koeficient. Modularita v Gephi vychází ze článku [23]. Tuto modularitu budeme v následující kapitole porovnávat s modularitou, která vyšla nám.

Gephi má, bohužel, i omezení. A to omezení na přidělenou paměť. Chceme-li pracovat s většími grafy - řádově o 100 000 hranách a více, musíme pamatovat na přidělení více paměti Javě. Gephi u více jak milionu vrcholů a hran doporučuje dokonce 8GB paměti [17]. V případě větších grafů tedy doporučujeme použít výkonnější počítač. Z tohoto důvodu jsme vybrali takové datové kolekce, které zvládneme zobrazit na osobním notebooku.

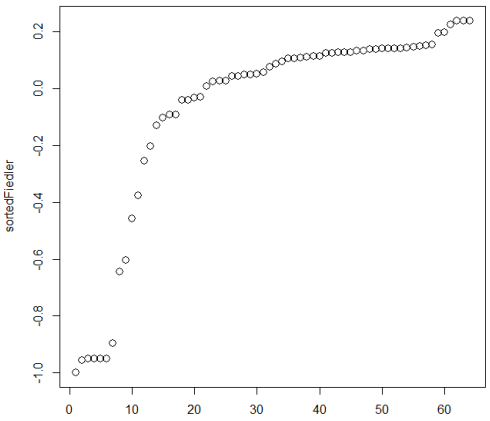
6.2 Kontrola výsledků v jazyce R

R je volně dostupný programovací jazyk, s možností použití uživatelského rozhraní, určený pro statistické, výpočetní problémy. Umožňuje grafickou vizualizaci výsledků velké škály statistických a grafických technik jako jsou například: lineární a nelineární modelování, statistické testy, analýza časových řad, klasifikace, shlukování atd.[27].

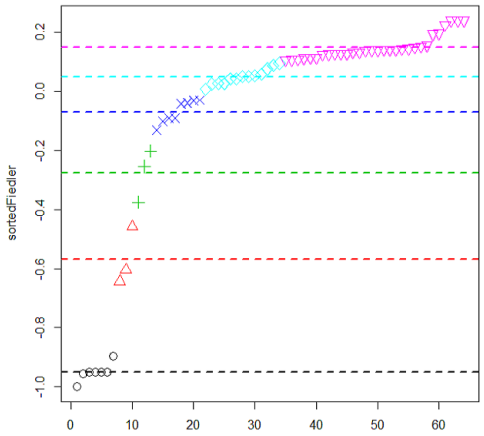
V této diplomové práci byl jazyk R použit pro kontrolu výpočtů vlastních vektorů Laplaceových matic a následnou vizualizaci tzv. Fiedlerova vektoru pro lepší představu o správnosti výpočtů spektrálních algoritmů za použití dříve zmiňovaných knihoven a vlastních implementací. Ve fázi, kdy nebyly naiplemenotovány metody měřící kvalitu, bylo možné posoudit výsledky jen vizuálně pomocí programu Gephi. Právě díky vizualizaci Fiedlerova vektoru můžeme přibližně určit, do kterých shluků by měly které vrcholy grafu patřit.

V našem programu bylo možno vypsat z matice U vlastní vektor příslušný druhému nejmenšímu vlastnímu číslu pomocí metody *public static string GetMatrixUInfo(double[,] matrixU)*, jež je součástí třídy *SpectralClustering* komponenty *GraphManipulation*, a ten poté setřídit a vizualizovat pomocí uživatelského rozhraní jazyka R. Na obrázku 10 můžeme vidět setříděný Fiedlerův vektor matice L_{rw} datové kolekce obsahující teroristy, která byla zmíněna v kapitole 5.2 a na obrázku 11 vrcholy přiřazené do shluků na základě hodnot Fiedlerova vektoru. Na obrázku 12 je možno shlédnout tento graf při použití nenormovaného algoritmu za použití šesti shluků a šesti

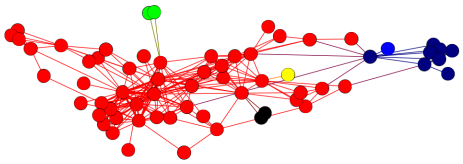
vlastních vektorů. Obrázek 13 znázorňuje stejný graf se stejným počtem shluků a vlastních vektorů avšak analyzován pomocí normovaného spektrálního algoritmu Shi a Malika[9].



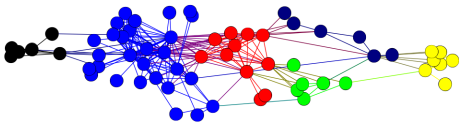
Obrázek 10: Setříděný Fiedlerův vektor datasetu s teroristy



Obrázek 11: Setříděný Fiedlerův vektor datasetu s teroristy po shlukování



Obrázek 12: Dataset s teroristy po použití nenormovaného spektrálního algoritmu - 6 shluků, 6 vlastních vektorů



Obrázek 13: Dataset s teroristy po použití normovaného spektrálního algoritmu Shi a Malika - 6 shluků, 6 vlastních vektorů

Abychom dosáhli požadovaných výsledků v jazyce R, byl použit následující zdrojový kód:

```
fiedlerVector <- c(0.109741,0.115098, ... ,-0.643366)
sortedFiedler <- sort(fiedlerVector)
plot(sortedFiedler, pch=1, cex=1.4)
k <- 6
result <- Ckmeans.1d.dp(sortedFiedler, k)
plot(sortedFiedler, col=result\$cluster, pch=result\$cluster, cex=1.5,
sub=paste("Number of clusters given:", k))
abline(h=result\$centers, col=1:k, lty="dashed", lwd=2)
```

Výpis 3: Ukázka zdrojového kódu jazyka R pro vizualizaci Fiedlerova vektoru

Na prvním řádku jsme definovali Fiedlerův vektor a na druhém řádku ho seřadili. Následující řádek setříděný vektor pomocí funkce *plot* vykreslil na plátno. Dalším krokem byla definice počtu

[9] Analýza této datové kolekce je v kapitole 6.4. Vizualizaci této datové kolekce s odlišnými parametry zobrazují obrázky 26, 27 a 31, které jsou v podkapitole A.1 přílohy A.

shluků a následné shlukování bodů tohoto jednorozměrného pole pomocí balíčku Ckeamns.1d.dp [28], který musel být zvlášť stažen, jelikož není součástí základní verze uživatelského rozhraní pro jazyk R a slouží pro jedno-dimenzionální k-means shlukování.

6.3 Cíle experimentů nad grafovými datovými kolekcemi

V následujících kapitolách se budeme věnovat experimentům s daty, které obsahují vrcholy a hrany, a mohou být rovnou ukládány do vážené matice sousednosti W. Data budeme postupně analyzovat algoritmy využívajících spektrálních rozkladů a vrcholy shlukovat do k shluků. Původní spektrálně shlukovací algoritmy pracují s variantou, kde *k shluků = k vlastních vektorů*. Nás bude zajímat, zda má vliv měnit i počet vlastních vektorů. Zda můžeme u některých algoritmů redukovat dimenzi tím, že pro nějaké číslo k shluků zvolíme nižší číslo l vlastních vektorů. Na základě výsledků měřících metod budeme vyhodnocovat, jaké k a l jsou pro daný graf a algoritmus nejlepší. U algoritmů bude měřen i porovnáván čas výpočtů. Grafy budou poté vizualizovány programem Gephi, který umožňuje rozdělit vrcholy grafu do shluků na základě výpočtu modularity. Dalším experimentem bude porovnání rozdělení vrcholů grafu programem Gephi a algoritmy využívajících spektrálních rozkladů. V tomto případě budeme procházet a analyzovat soubor GraphML, který vygeneruje program Gephi a počítat pro něj konduktanci a cut ratio. Posledním experimentem bude porovnání jednotlivých měřících metod.

Vysvětlivky k následujícím tabulkám:

1. Shl. - počet shluků pro k-means,
2. Vl. ve. - počet vlastních vektorů matice U, případně T,
3. Mod. - námi vypočítaná modularita,
4. Kon. - námi vypočítaná konduktance,
5. CutRat. - námi vypočítané cut ratio,
6. D.I. - námi vypočítaný dunnův index,
7. S.I. - námi vypočítaný silhouette index,
8. Čas - celkový čas výpočtu ve formátu HH:MM:SS.MS

6.3.1 Jak boudou experimenty probíhat

Nejprve budou vypočítány všechny potřebné matice pro daný algoritmus. Následně budou spuštěny dva cykly. Jeden, ve kterém se bude navyšovat počet shluků a druhý, kde bude navyšován počet vlastních vektorů. Na základě aktuálního nastavení budou vypočítány pro daný graf hodnoty měřících indexů a společně s časem výpočtu budou uloženy do databáze. Na základě výsledků vybereme 3 výsledky s nejvyšší modularitou a 3 výsledky s vysokou modularitou a nízkou

konduktancí a cut ratiem. Poté přidáme jeden výsledek pro porovnání, kde je počet shluků roven počtu vlastních vektorů, nejlepší výsledek pro stejný počet shluků, který vypočítal program Gephi a jeden nejlepší výsledek pouze s jedním vlastním vektorem. Vizualizovat budeme 2 nejlepší výsledky a výsledek obarvený na základě výpočtu modularity programem Gephi. Výsledky se nachází v následujících kapitolách, případně v příloze A.

6.4 Datová kolekce z univerzity Koblenz obsahující teroristy

Jako první bude analyzována datová kolekce veřejně dostupná na stránkách university Koblenz [26], která obsahuje teroristy a jejich vazby. Graf obsahuje 64 vrcholů a 243 hran. Vizualizace tohoto grafu byla uvedena v předchozí kapitole 6.2 a je na obrázcích 12, 13 a v podkapitole A.1 přílohy A poté na obrázcích 26 až 31. Tabulky 2, 3 a 4 znázorňují analýzy datasetu spektrálními algoritmy. Hodnoty na základě rozdělení vrcholů do 6 shluků programem ukazuje tabulka 1.

Tabulka 1: Hodnoty datasetu s teroristy po rozdělení do 6 shluků programem Gephi.

Modularita	Konduktance	CutRatio
0.4352	0.2341	0.0296

Tabulka 2: Teroristi - Nenormovaný algoritmus (L)

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S. I.	Čas
6	2	0.3799	**0.2618**	0.0352	2.55e-16	0.1184	00:00:00.061
7	1	0.3521	0.3837	0.0443	-	0.0463	00:00:00.055
7	3	**0.3952**	0.3547	0.0242	4.33e-16	0.1032	00:00:00.071
11	4	0.3783	0.5395	0.0266	**5.04e-16**	0.1907	00:00:00.071
12	7	0.3755	0.5367	0.0261	3.76e-16	0.1809	00:00:00.073
13	11	0.3902	0.6925	0.0208	4.40e-16	0.1635	00:00:00.080
14	9	0.3755	0.6743	0.0245	4.18e-16	0.1833	00:00:00.078
14	14	0.2552	0.6878	**0.0199**	4.79e-16	**0.3207**	00:00:00.084

Tabulka 3: Teroristi - Normovaný algoritmus Shi a Malika (L_{rw})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S. I.	Čas
4	3	0.4077	**0.1497**	**0.0259**	4.77e-17	0.2462	00:00:00.11
5	3	0.4028	0.1993	0.0349	4.77e-17	0.2375	00:00:00.11
5	5	0.4008	0.1904	0.0316	7.20e-17	0.2602	00:00:00.13
6	4	0.4027	0.2341	0.0333	7.59e-17	0.3155	00:00:00.12
7	1	0.3549	0.2926	0.0504	-	-0.0557	00:00:00.09
7	8	0.4062	0.2515	0.0364	1.73e-16	0.2525	00:00:00.16
8	6	**0.4092**	0.2681	0.0333	**8.91e-16**	**0.3761**	00:00:00.12
9	8	0.4068	0.2837	0.0358	1.89e-16	0.3204	00:00:00.14

Tabulka 4: Teroristi - Normovaný algoritmus Ng, Jordana a Weisse (L_{sym})

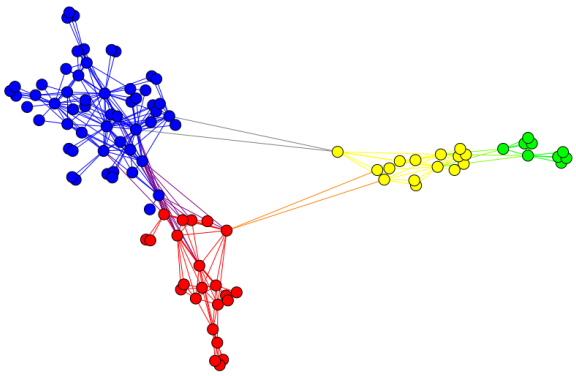
Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S. I.	Čas
5	19	0.3805	**0.2824**	0.0349	**1.91e-15**	-0.0171	00:00:00.11
6	14	0.3793	0.3039	0.0327	4.49e-016	-0.0674	00:00:00.11
10	13	0.3856	0.6153	0.0276	4.31e-16	-0.0391	00:00:00.12
12	13	**0.3901**	0.6297	0.0288	1.33e-15	-0.0105	00:00:00.13
12	14	0.3803	0.5996	0.0312	4.49e-16	-0.0508	00:00:00.11
13	13	0.3837	0.6598	0.0275	4.31e-16	-0.0173	00:00:00.12
14	19	0.3832	0.6928	**0.0231**	7.18e-16	0.0948	00:00:00.19
17	19	0.3856	0.6245	0.0279	7.18e-16	**0.1046**	00:00:00.14

6.5 Experimenty nad syntetickými sítěmi

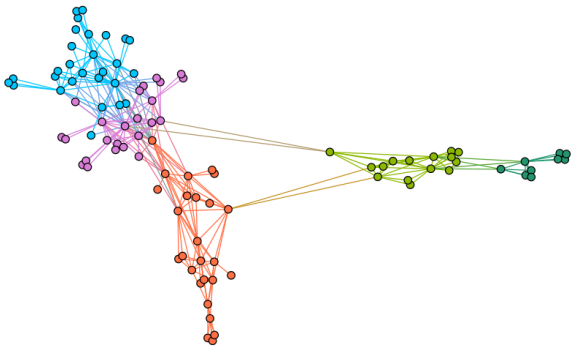
Následující podkapitoly se věnují analýze různě rozsáhlým syntetickým sítím, které poskytl pan doc. Mgr. Miloš Kudělka, Ph.D.

6.5.1 Datová kolekce D_102_318

Datová kolekce obsahuje 102 vrcholů a 318 hran. Naměřené hodnoty na základě rozdělení vrcholů do 5 shluků programem Gephi ukazuje tabulka 5. Tabulky 6, 7 a 8 zobrazují naměřené hodnoty pro různé počty shluků a vlastních vektorů odlišnými algoritmy. Obrázek 14 zobrazuje graf rozdělen do 4 shluků na základě 4 vlastních vektorů algoritmem Shi a Malika. Obrázek 15 ukazuje rozdělení vrcholů do 5 shluků na základě výpočtu modularity programem Gephi.



Obrázek 14: Datová kolekce **D_102_318** - Nenormovaný algoritmus Shi a Malika. 4 shluky a 4 vlastní vektory



Obrázek 15: Datová kolekce **D_102_318** - Vrcholy rozděleny do 5 shluků programem Gephi.

Tabulka 5: Hodnoty sítě **D_102_318** po rozdělení do **5** shluků programem Gephi.

Modularita z Gephi	Konduktance	CutRatio
0.589	0.1357	0.0126

Tabulka 6: Datová kolekce **D_102_318** - Nenormovaný algoritmus (L)

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S. I.	Čas
5	1	0.4699	0.1866	0.0189	0.0002	-0.2575	00:00:00.18
5	3	0.5407	**0.1312**	0.0112	0.0016	**0.3676**	00:00:00.22
7	7	0.4962	0.1916	0.0115	0.0102	0.2304	00:00:00.31
9	5	0.5403	0.2224	0.0122	0.0141	0.2989	00:00:00.21
10	6	0.5413	0.3009	0.0142	0.0233	0.2831	00:00:00.23
13	10	**0.5513**	0.3992	**0.0106**	**0.0226**	0.3425	00:00:00.39
14	11	0.5509	0.3949	**0.0106**	**0.0226**	0.3516	00:00:00.41
15	12	0.5485	0.4356	0.0107	0.0225	0.3536	00:00:00.44

Tabulka 7: Datová kolekce **D_102_318** - Normovaný algoritmus Shi a Malika (L_{rw})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S. I.	Čas
4	4	0.5434	**0.0795**	**0.0069**	0.0041	0.3299	00:00:00.33
4	8	0.5408	0.1169	0.0076	0.0181	0.1961	00:00:00.49
4	6	0.5408	0.1169	0.0076	0.0137	0.2444	00:00:00.40
5	1	0.4711	0.1865	0.0189	0.0002	-0.2857	00:00:00.17
5	5	0.5491	0.1201	0.0078	0.0143	0.3504	00:00:00.37
6	4	0.5573	0.1439	0.0098	0.0042	0.3671	00:00:00.30
6	5	0.5573	0.1439	0.0098	0.0167	**0.3784**	00:00:00.37
7	6	**0.562**	0.1637	0.0099	0.0165	0.3387	00:00:00.38
8	6	0.5599	0.1897	0.0127	0.0165	0.3147	00:00:00.30

Tabulka 8: Kolekce **D_102_318** - Normovaný algoritmus Ng, Jordana a Weisse (L_{sym})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S.I.	Čas
3	**3**	0.5481	**0.0596**	**0.0061**	0.0005	**0.2901**	00:00:00.20
3	8	0.5481	**0.0596**	**0.0061**	**0.0026**	0.1451	00:00:00.49
3	9	0.5457	0.0623	0.0064	0.0038	0.1094	00:00:00.51
5	14	0.5461	0.1465	0.0107	0.0051	0.0871	00:00:00.51
6	17	**0.5564**	0.1835	0.0154	0.0051	0.0131	00:00:00.55
7	13	0.5542	0.2197	0.0181	0.0051	0.0634	00:00:00.45
8	17	0.5543	0.2278	0.0183	0.0051	0.0267	00:00:00.55
9	17	0.5555	0.2429	0.0184	0.0051	0.0511	00:00:00.56
9	13	0.5531	0.2475	0.0191	0.0051	0.1059	00:00:00.53

6.5.2 Datová kolekce D_100_685

Datová kolekce obsahuje 100 vrcholů a 685 hran. Tabulka 9 znázorňuje hodnoty po rozdělení grafu do 7 shluků progamem Gephi. Hodnoty spektrálních algoritmů jsou v tabulkách 10, 11 a 12. Obrázek 16 zobrazuje graf rozdělen do 7 shluků na základě 4 vlastních vektorů algoritmem Shi a Malika a byl vybrán jako nejlepší rozdělení dle naměřených hodnot. Obrázek 17 ukazuje rozdělení vrcholů do 7 shluků po výpočtu modularity programem Gephi.

Tabulka 9: Hodnoty datasetu **D__100__685** po rozdělení do **7** shluků programem Gephi.

Modularita	Konduktance	CutRatio
0.2668	0.3548	0.0828

Tabulka 10: Datová kolekce **D__100__685** - Nenormovaný algoritmus (L)

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S. I.	Čas
7	1	0.1208	**0.3832**	0.1069	-	-0.7311	00:00:00.13
13	13	0.0715	0.4196	**0.0217**	**5.95e-16**	-0.4153	00:00:01.10
21	11	**0.1681**	0.4448	0.0498	1.83e-16	-0.3797	00:00:00.30
28	17	0.1423	0.4663	0.0543	5.62e-16	-0.2886	00:00:00.36
29	17	0.1446	0.4691	0.0595	5.62e-16	**-0.2803**	00:00:00.38

Tabulka 11: Datová kolekce **D__100__685** - Normovaný algoritmus Shi a Malika (L_{rw})

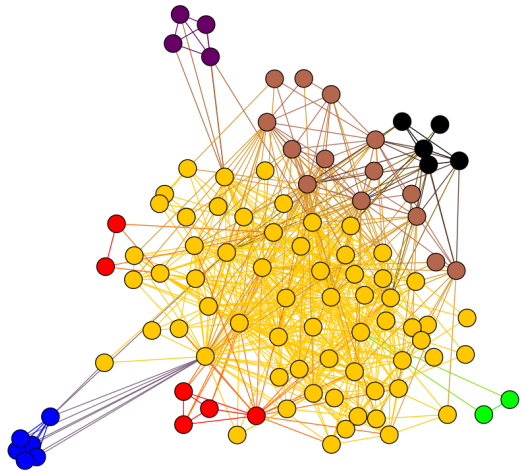
Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S. I.	Čas
7	1	0.1221	0.373	0.1111	-	-0.6845	00:00:00.21
7	4	0.1511	**0.2913**	**0.0421**	4.07e-17	**-0.0604**	00:00:00.31
8	5	0.1786	0.3078	0.0434	6.10e-17	-0.1158	00:00:00.31
10	8	0.209	0.3258	0.0409	6.45e-17	-0.3108	00:00:00.31
10	11	0.2084	0.3329	0.0472	**6.79e-17**	-0.2022	00:00:00.41
11	9	0.2141	0.3348	0.0454	5.55e-17	-0.2703	00:00:00.35
12	8	**0.2173**	0.3523	0.0596	6.45e-17	-0.2796	00:00:00.30
12	12	0.1993	0.3469	0.0486	6.63e-17	-0.1971	00:00:00.33
16	12	0.2011	0.3769	0.0613	6.63e-17	-0.2066	00:00:00.32

Tabulka 12: Kolekce **D__100__685** - Normovaný algoritmus Ng, Jordana a Weisse (L_{sym})

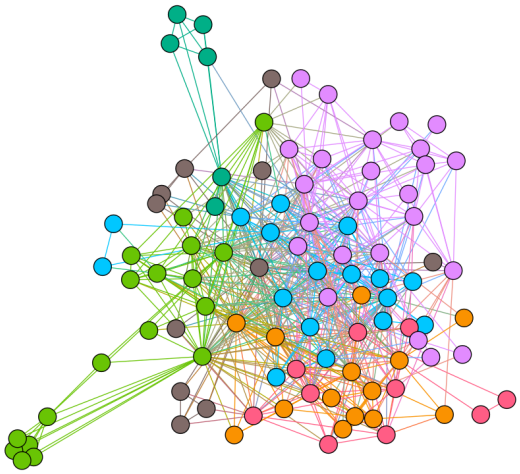
Shl.	Vl. ve.	Mod.	Kon.	CutRat.	D.I.	S.I.	Čas
7	31	0.1322	**0.3621**	0.0647	1.16e-15	**0.0323**	00:00:01.92
12	31	0.1565	0.4005	0.0648	1.16e-15	-0.3895	00:00:01.51
12	34	0.1545	0.4107	0.0771	**4.60e-15**	-0.4116	00:00:01.53
13	15	**0.1937**	0.4119	0.0708	4.47e-16	-0.4867	00:00:00.92
13	22	0.1715	0.4129	0.0607	6.64e-16	-0.3932	00:00:01.21
14	32	0.1552	0.4149	0.0677	1.31e-15	-0.3861	00:00:01.24
18	12	0.1736	0.4401	0.0667	4.09e-16	-0.5478	00:00:00.89
20	16	0.1693	0.4608	0.0638	8.86e-16	-0.4791	00:00:00.96
29	29	0.1631	0.4843	**0.0488**	1.08e-15	-0.2787	00:00:01.22

6.5.3 Datová kolekce D__501__1502

Datová kolekce obsahuje 501 vrcholů a 1502 hran. Rozdělení do 15 shluků programem Gephi zobrazuje obrázek 19 a hodnoty pak tabulka 13. Tabulky 14, 15 a 16 zobrazují hodnoty pro spektrální algoritmy. U tohoto grafu, z důvodu časové náročnosti výpočtu, nebudou počítány silhouette index a dunnův index. Obrázek 18 zobrazuje graf rozdělený do 16 shluků na základě 12



Obrázek 16: Datová kolekce **D__100__685** - normovaný algoritmus Shi a Malika. 7 shluků a 4 vlastní vektory



Obrázek 17: Datová kolekce **D__100__685** - rozdělení do 7 shluků programem Gephi.

vlastních vektorů algoritmem Shi a Malika a byl vybrán jako nejlepší rozdělení dle naměřených hodnot. Další vizualizaci grafu můžeme nalézt na obrázku 34 v podkapitole A.2 přílohy A, který je rozdělený do 24 shluků na základě 23 vlastních vektorů.

Tabulka 13: Hodnoty datasetu **D__501__1502** po rozdělení do **15** shluků programem Gephi.

Modularita	Konduktance	CutRatio
0.666	0.145	0.002

Tabulka 14: Datová kolekce **D__501__1502** - Nenormovaný algoritmus (L)

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
15	7	0.5522	**0.1472**	**0.0019**	00:00:01.73
18	1	0.4208	0.3775	0.0069	00:00:01.70
25	19	0.5966	0.2205	0.0021	00:00:01.77
26	19	0.6017	0.1886	0.0021	00:00:01.75
27	11	**0.6036**	0.2327	0.0033	00:00:01.74
30	30	0.5351	0.2612	**0.0019**	00:00:01.83
33	20	0.5967	0.2436	0.0025	00:00:01.72

6.5.4 Datová kolekce D__501__2887

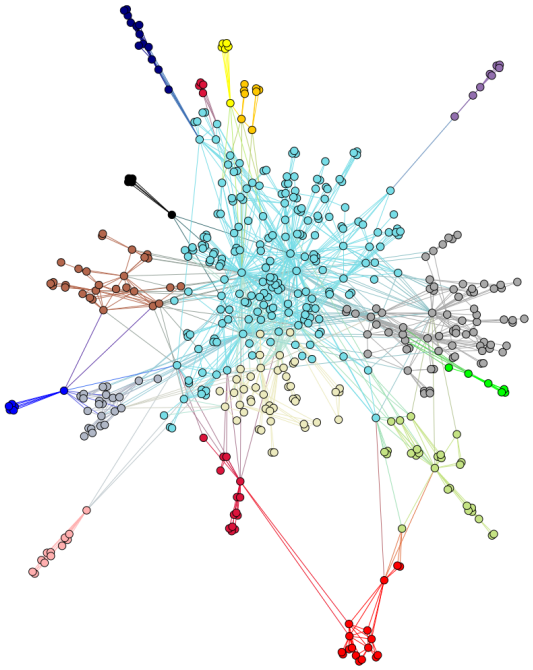
Datová kolekce obsahuje 501 vrcholů a 2887 hran. Program Gephi ji rozdělil do 12 shluků, hodnoty zobrazuje tabulka 17 a vizualizace je na obrázku 21. Naměřené hodnoty pro spektrální algoritmy jsou v tabulkách 18, 19 a 20. Pro tento graf nebude počítán silhouette index ani dunnův index. Obrázek 20 zobrazuje graf rozdělený do 24 shluků na základě 23 vlastních vektorů algoritmem Shi a Malika a byl vybrán jako nejlepší rozdělení dle naměřených hodnot.

Tabulka 15: Datová kolekce **D__501__1502** - Normovaný algoritmus Shi a Malika (L_{rw})

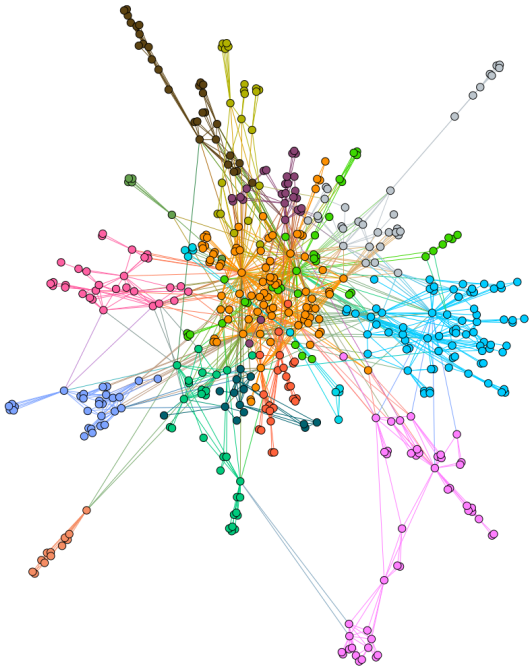
Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
15	10	0.5896	0.1291	0.0018	00:00:05.53
16	1	0.4371	0.3346	0.0065	00:00:05.51
16	12	0.6007	**0.1213**	**0.0016**	00:00:05.55
20	20	0.6143	0.1324	0.0017	00:00:05.57
24	23	**0.6304**	0.1501	0.0019	00:00:05.58
25	21	0.6289	0.1533	0.0021	00:00:05.59
32	32	0.6288	0.1723	0.0021	00:00:05.62

Tabulka 16: Kolekce **D__501__1502** - Normovaný algoritmus Ng, Jordana a Weisse (L_{sym})

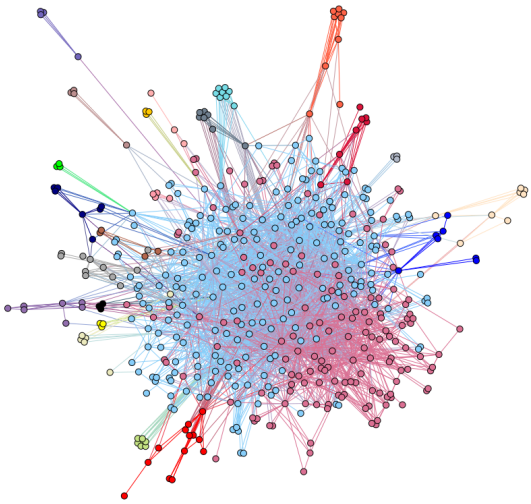
Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
11	22	0.6597	**0.1541**	**0.0024**	00:00:01.15
11	24	0.6574	0.1623	0.0025	00:00:01.15
11	27	0.6511	0.1633	0.0025	00:00:01.80
13	28	0.6554	0.1641	0.0025	00:00:01.15
13	32	0.6636	0.1701	0.0027	00:00:01.82
15	32	**0.6674**	0.1863	0.0031	00:00:01.82
18	18	0.6312	0.2151	0.0032	00:00:01.15
19	32	0.6651	0.2591	0.0035	00:00:01.82



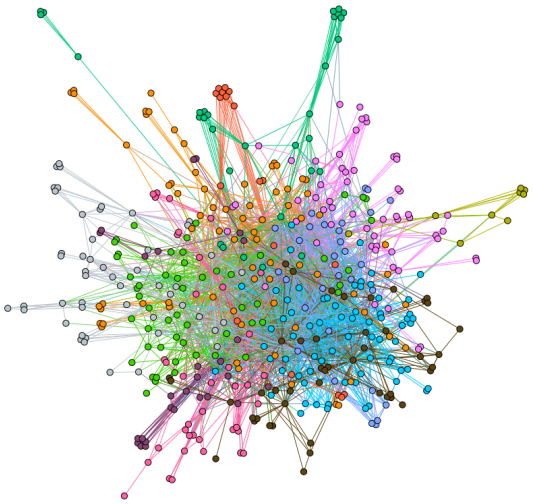
Obrázek 18: Datová kolekce **D__501__1502** - Normovaný algoritmus Shi a Malika. 16 shluků a 12 vlastních vektorů.



Obrázek 19: Datová kolekce **D__501__1502**. Vrcholy rozděleny do 16 shluků na základě výpočtu modularity pomocí programu Gephi.



Obrázek 20: Datová kolekce **D__501__2887** - Normovaný algoritmus Shi a Malika. 24 shluků a 23 vlastních vektorů.



Obrázek 21: Datová kolekce **D__501__2887**. Vrcholy rozděleny do 12 shluků programem Gephi.

Tabulka 17: Hodnoty datasetu **D__501__2887** po rozdělení do 12 shluků programem Gephi.

Modularita	Konduktance	CutRatio
0.3803	0.3291	0.0121

Tabulka 18: Datová kolekce **D__501__2887** - Nenormovaný algoritmus (L)

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
10	1	0.1546	**0.3495**	0.0225	00:00:02.02
12	2	0.1616	0.3835	0.0126	00:00:02.03
77	54	0.2437	0.4975	0.0051	00:00:02.20
91	51	0.2357	0.4991	0.0075	00:00:02.40
91	91	0.1569	0.4854	**0.0037**	00:00:02.51
97	49	**0.2374**	0.5032	0.0077	00:00:02.41

Tabulka 19: Datová kolekce **D__501__2887** - Normovaný algoritmus Shi a Malika (L_{rw})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
12	1	0.1457	0.3731	0.0232	00:00:05.82
24	23	0.2841	**0.2516**	**0.0043**	00:00:05.91
29	28	0.2569	0.2668	**0.0043**	00:00:05.93
30	30	0.2563	0.2721	0.0045	00:00:06.01
43	29	**0.3028**	0.3446	0.0061	00:00:06.01
47	33	0.2924	0.3297	0.0062	00:00:06.14
93	49	0.2547	0.4011	0.0097	00:00:07.10

Tabulka 20: Kolekce **D__501__2887** - Normovaný algoritmus Ng, Jordana a Weisse (L_{sym})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
12	100	0.2595	0.3293	0.0121	00:00:02.36
17	94	0.2707	**0.3301**	0.0099	00:00:02.34
20	101	0.2741	0.3667	0.0109	00:00:02.74
38	104	0.2805	0.3938	0.0095	00:00:03.56
45	50	**0.2878**	0.4292	**0.0077**	00:00:02.11
66	66	0.2649	0.4453	0.0077	00:00:02.54
78	68	0.2811	0.4606	0.0086	00:00:02.58

6.5.5 Datová kolekce D__2000__5700

Datová kolekce obsahuje 2000 vrcholů a 5700 hran a programem Gephi byla rozdělena do 12 shluků. Tabulka 21 znázorňuje pro toto rozdělení hodnoty. Výsledky spektrálních algoritmů jsou v tabulkách 22, 23 a 24. Silhouette index a dunnův index nebudou počítány. Obrázek 35, který se nachází v příloze A, zobrazuje graf rozdělený do 31 shluků na základě 26 vlastních vektorů algoritmem Shi a Malika.

Tabulka 21: Hodnoty sítě **D__2000__5700** po rozdělení do 31 shluků programem Gephi.

Modularita	Konduktance	CutRatio
0.784	0.1039	0.0003

Tabulka 22: Datová kolekce **D__2000__5700** - Nenormovaný algoritmus (L)

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
23	1	0.6071	0.3074	0.0009	00:01:40.27
31	13	0.7223	0.1696	0.0005	00:01:41.64
36	23	0.7533	**0.1148**	0.0003	00:02:03.03
44	44	0.7334	**0.1148**	**0.0002**	00:02:59.81
46	23	**0.7614**	0.1595	0.0004	00:02:51.51
58	46	0.7515	0.1424	0.0003	00:03:01.11
72	37	0.7589	0.2196	0.0006	00:02:59.71

Tabulka 23: Datová kolekce **D__2000__5700** - Normovaný algoritmus Shi a Malika (L_{rw})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
29	29	0.7242	**0.0779**	**0.0002**	00:06:31.61
31	1	0.5722	0.3498	0.0017	00:06:26.86
31	26	0.7607	0.0821	**0.0002**	00:06:27.72
32	26	0.7626	0.0941	0.0003	00:06:27.69
38	29	0.7644	0.1067	0.0003	00:06:27.87
39	28	**0.7659**	0.1097	0.0004	00:06:27.97
40	28	0.7601	0.1013	0.0003	00:06:27.75

Tabulka 24: Kolekce **D__2000__5700** - Normovaný algoritmus Ng, Jordana a Weisse (L_{sym})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
17	32	0.7808	0.1116	0.0004	00:02:10.59
18	37	0.7809	**0.1035**	**0.0003**	00:02:10.73
22	36	0.7808	0.1087	**0.0003**	00:02:10.78
31	79	0.7818	0.1295	0.0004	00:02:12.91
34	30	0.7817	0.1305	0.0004	00:02:11.98
34	34	0.7809	0.1385	0.0004	00:02:10.91
36	79	0.785	0.1277	0.0004	00:02:13.01
37	79	**0.7858**	0.1315	0.0004	00:02:12.02
100	100	0.7332	0.2087	0.0007	00:02:14.97

6.5.6 Datová kolekce D__2000__12306

Datová kolekce obsahuje 2000 vrcholů a 12306 hran. Programem Gephi byla rozdělena do 19 shluků. Hodnoty pro toto rozdělení zobrazuje tabulka 25. Graf byl analyzován pouze algoritmem Shi a Malika a opět nebude počítán silhouette index a dunnův index. Naměřené hodnoty jsou v tabulce 26. V příloze A lze vidět tuto datovou kolekci zvizualizovanou na obrázku 36.

Tabulka 25: Hodnoty sítě **D__2000__12306** po rozdělení do **19** shluků programem Gephi.

Modularita	Konduktance	CutRatio
0.4232	0.3121	0.0029

Tabulka 26: Datová kolekce **D__2000__12306** - Normovaný algoritmus Shi a Malika (L_{rw})

Shl.	Vl. ve.	Mod.	Kon.	CutRat.	Čas
19	1	0.1601	0.3611	0.0043	00:07:44.01
40	40	0.1141	**0.2107**	**0.0008**	00:07:45.90
49	37	0.2548	0.2461	0.0012	00:07:45.37
55	49	0.2515	0.2484	0.0011	00:07:46.20
81	48	**0.3037**	0.3006	0.0015	00:07:48.55
89	43	0.2917	0.3153	0.0021	00:07:46.78
91	49	0.2881	0.3148	0.0018	00:07:47.85

6.6 Shrnutí experimentů nad datovými kolekcemi

V uvedených experimentálních tabulkách můžeme vypozorovat, že má vliv vkládat odlišný počet vlastních vektorů od počtu shluků u každého ze tří algoritmů. Každý algoritmus se chová v této problematice odlišně. Zatímco u nenormovaného algoritmu a algoritmu od Shi a Malika vidíme, že byly výsledky přesnější při menším počtu vlastních vektorů než shluků, u normovaného algoritmu Ng, Jordana a Weisse to platí obráceně.

Jako nejpřesnější algoritmus byl vybrán algoritmus Shi a Malika, který využívá normovanou Laplaceovu matici L_{rw}. Tento algoritmus ve většině případů dosahoval nejvyšší modularity zároveň s nejnižší konduktancí a ratio cutem. U kolekce D_2000_5700 dosahoval nejvyšší modularity normovaný algoritmus Ng, Jordana a Weisse, který používá Laplaceovu matici L_{sym}. Konduktance i ratio cut však vycházel opět lépe pro algoritmus Shi a Malika.

V uvedených časech vidíme, že i na úkor přesnosti trvá algoritmus Shi a Malika déle, než ostatní dva algoritmy. Matice L_{rw} je nesymetrická a proto knihovna *Alglib* počítá vlastní čísla a vlastní vektory z této matice déle. Další faktor hrající roli v časech výpočtu je počet vlastních vektorů. Proto je pozitivní, že u datové kolekce D_100_685 byl nejlepší výsledek, i na úkor nižší modularity, pro 7 shluků na základě 4 vlastních vektorů. U datové kolekce D_2000_12306 vyšla nejvyšší modularita pro 81 shluků a 48 vlastních vektorů, což je opět téměř poloviční redukce.

Co se týče rozdělení vrcholů grafů do shluků na základě výpočtu modularity programem Gephi, lze říci, že algoritmy využívající spektrálních rozkladů Laplaceových matic dosahují lepších výsledků i na úkor nižší modularity. Modularita je sice nižší, avšak indexy konduktance a ratio cut vycházely lépe pro spektrální algoritmy.

Proto lze učinit i závěr, že bychom se neměli spoléhat pouze na modularitu, která sice s rostoucí hodnotou může vypovídat o lepším rozdělení vrcholů grafů do shluků, ale ne ve všech případech. Tabulky znázorňují případy, kde rozdělení s nepatrně nižší modularitou dosahují lepších výsledků na základě ostatních indexů.

Kvalita a chování indexů jsou odlišné pro různě rozsáhlé kolekce a pro odlišné počty shluků. Nemůžeme prohlásit, že existuje „nejlepší“ měřící metrika pro shlukování na grafech, a tedy se nemůžeme spoléhat pouze na jednu.

6.7 Experimenty nad obrázky

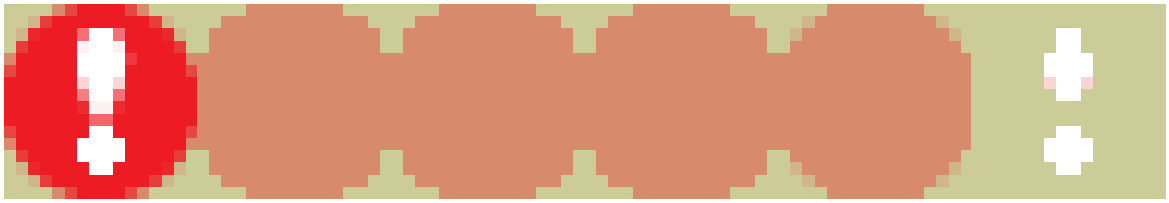
V následujících podkapitolách budou probírány experimenty se segmentací obrazu za použití spektrálních algoritmů nad různými obrázky.

6.7.1 Cíle experimentů

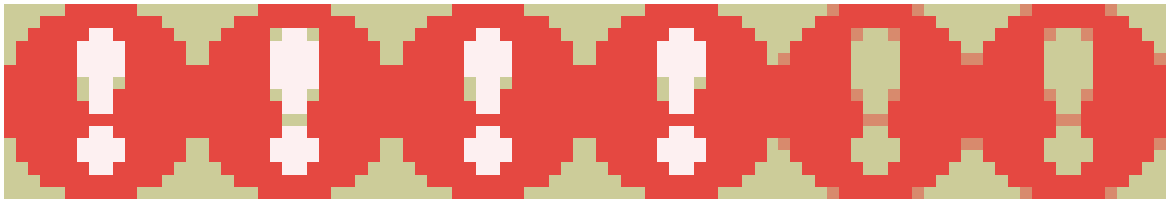
Cílem experimentů bude zkoumat chování parametrů *sigma* a *treshold*, které slouží jako vstupní parametry pro konstrukci vážené matice sousednosti W. Tyto parametry budou postupně měněny a testovány různými algoritmy na různý počet shluků a vlastních vektorů. Z předchozí kapitoly víme, jak se který algoritmus chová pro různé počty shluků a vlastních vektorů, a proto budou tyto informace brány v potaz. Dalšími cíli bude porovnání velikosti originálního obrázku s velikosti obrázku po shlukování a náročnost na paměť různě řídkých matic. U experimentů s obrázky nebudou uváděny časy výpočtů, protože jsme v rámci efektivnějších výpočtů použili knihovnu Alglib pro výpočet většího množství vlastních vektorů a poté se měnily počty použitých vlastních vektorů.

6.7.2 Parametr sigma

V první řadě budeme experimentovat pouze s parametrem *sigma* a jakmile zjistíme, jaký nejvhodnější můžeme použít pro obrázky, přejdeme k experimentům parametru *treshold*. V předchozí sekci byl jako nejpřesnější algoritmus vyhodnocen algoritmus Shi a Malika, proto na testování použijeme tento algoritmus.



(a) Originál (b) 1000 (c) 100 (d) 10 (e) 0.9 (f) 0.09



(g) 0.009 (h) 0.0009 (i) 0.00009 (j) 0.000009 (k) 0.0000009 (l) 0.00000009

Obrázek 22: Vizualizace obrázků po shlukování s odlišným parametrem sigma.

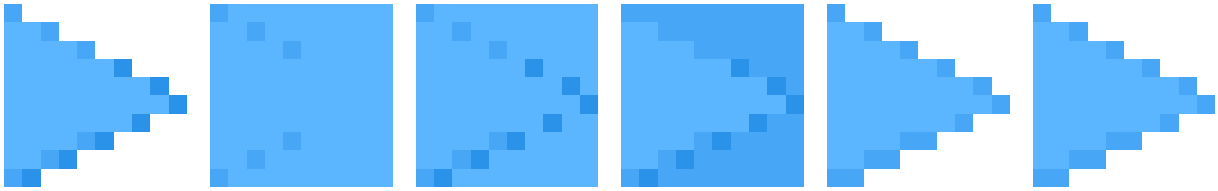
Na obrázku 22 můžeme vidět jak originální obrázek (22a), tak vizualizaci obrázku po shlukování s rozdílnými parametry *sigma* za použití 3 shluků a 1 vlastního vektoru. Z experimentu lze vyvodit, že čím vyšší *sigma*, tím horší výsledek. Proto jsme se snažili *sigmu* snižovat. Parametry na obrázcích 22g, 22i, 22j vrátily totožné výsledky, avšak méně přesné, než na obrázku 22h, který považujeme za nejpřesnější. Snižování parametru *sigma* vede opět ke zhoršujícím se výsledkům.

6.7.3 Parametr treshold

Na obrázku 23 vidíme různé varianty parametru *treshold*, které byly vygenerovány na základě 3 shluků, 1 vlastního vektoru a $sigma = 0.0009$ algoritmem Shi a Malika. Na základě výsledků na tomto malém obrázku budeme testovat treshold na větších obrázcích ve dvou variantách. U takto malého obrázku nevidíme rozdíl mezi variantami 23e a 23f. U větších obrázků budeme tedy experimentovat s dvěma parametry treshold $= 0.85$ a treshold $= 0.0009$ a porovnávat, jakou roli budou tyto dva parametry hrát.

6.7.4 Experimenty na obrázku rožnovského skanzenu

Zde provedeme experimenty na obrázku rožnovského skanzenu o velikosti 70x46 pixelů se zvoleným parametrem $sigma = 0.0009$. Nejprve budeme testovat parametr $treshold = 0.85$, poté $treshold = 0.0009$. Pro tyto dva případy zde budou vizualizovány výsledky algoritmu Shi a Malika.

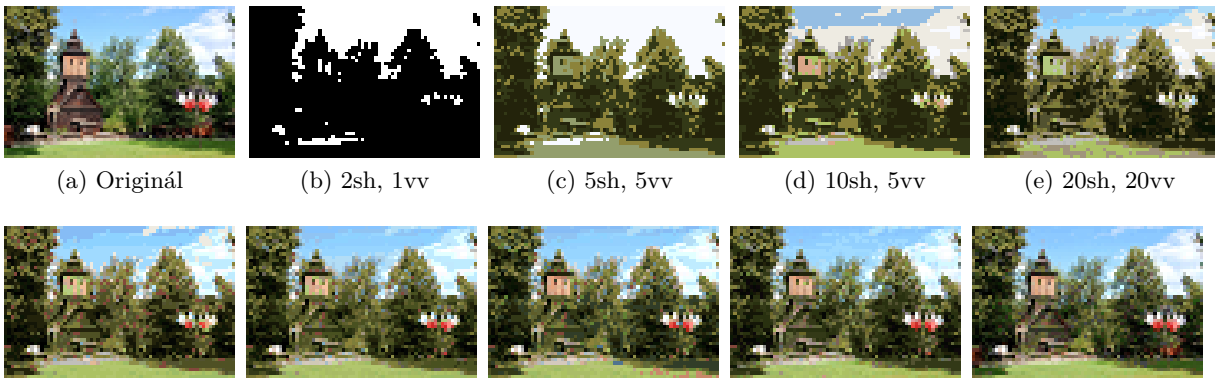


(a) Originál (b) 0.98 (c) 0.95 (d) 0.89 (e) 0.85 (f) 0.0009

Obrázek 23: Vizualizace obrázku po shlukování s odlišným parametrem treshold.

Výsledky pro nenormovaný algoritmus a normovaný algoritmus Ng, Jordana a Weisse nalezneme v příloze B.1. Popis obrázků značí nejprve počet shluků a poté počet vlastních vektorů.

6.7.4.1 Segmentace pro parametr sigma = 0.0009 a treshold = 0.85 Na obrázku 24 vidíme jak originální obrázek(24a), tak varianty obrázků pro různé počty shluků a vlastních vektorů. Tabulka 27 poté znázorňuje pro porovnání velikosti jak originálního obrázku, tak obrázků po segmentaci. Matice pro tento obrázek zabíraly v paměti 866 - 971MB v závislosti na velikosti matice U.



(a) Originál (b) 2sh, 1vv (c) 5sh, 5vv (d) 10sh, 5vv (e) 20sh, 20vv

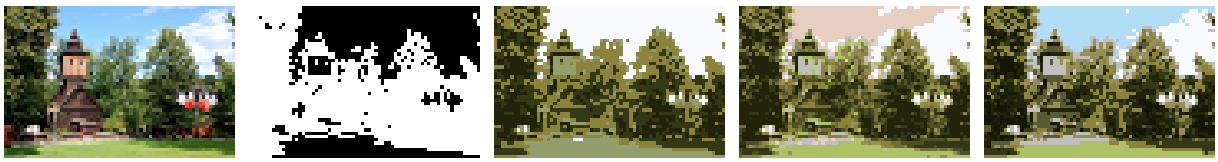
(f) 50sh, 50vv (g) 100sh, 50vv (h) 100sh, 100vv (i) 400sh, 400vv (j) 1000sh, 500vv

Obrázek 24: Vizualizace obrázku po shlukování s odlišnými počty shluků a vlastních vektorů s parametry sigma = 0.0009 a treshold = 0.85.

6.7.4.2 Segmentace pro parametr sigma = 0.0009 a treshold = 0.0009 Obrázek 25 zobrazuje opět jak originální obrázek (25a), tak různé počty shluků a vlastních vektorů. Díky nižšímu parametru treshold je segmentace přesnější, avšak matice hustší. Matice proto zabíraly v paměti 1121 - 1291MB v závislosti na velikosti matice U a jejich výpočet trval déle. Obrázky 25d, 25g, 25i, 25k, 25l, 25n a 25o zobrazují segmentaci pro různé počty shluků za použití pouze jednoho vlastního vektoru. Tabulka 28 poté obsahuje velikosti výsledků v bajtech. Můžeme si všimnout, že obrázek je 25j přesnější a menší, než obrázek 25i, který byl analyzován stejným počtem shluků avšak pouze jedním vlastním vektorem.

Tabulka 27: Velikosti obrázků v bajtech po shlukování z obrázku 24

Počet shluků	Počet vl. vektorů	Velikost
Originál	Originál	9 816 bajtů
2	1	572 bajtů
5	1	1190 bajtů
5	5	1423 bajtů
10	5	1645 bajtů
10	10	1783 bajtů
20	20	2337 bajtů
50	50	2970 bajtů
100	50	3632 bajtů
100	100	3590 bajtů
400	400	4840 bajtů
1000	500	7142 bajtů



(a) Originál (b) 2sh, 1vv (c) 5sh, 5vv (d) 10sh, 1vv (e) 10sh, 10vv

(f) 30sh, 30vv (g) 50sh, 1vv (h) 50sh, 50vv (i) 100sh, 1vv (j) 100sh, 100vv

(k) 500sh, 1vv (l) 1000sh, 1vv (m) 1000sh, 1000vv (n) 2000sh, 1vv (o) 3000sh, 1vv

Obrázek 25: Vizualizace obrázku po shlukování s odlišnými počty shluků a vlastních vektorů s parametry sigma = 0.0009 a treshold = 0.0009.

6.8 Shrnutí experimentů nad obrázky

V počáteční fázi experimentů byl vyhodnocen parametr *sigma* = 0.0009 jako nejlepší, a proto se dále experimentovalo s touto hodnotou. Parametr *treshold* byl použit ve dvou variantách (0.85 a 0.0009). Řidší matice, za použití tresholdu = 0.85, zabíraly méně paměti a výpočty byly rychlejší, avšak na úkor použití více shluků a tím spojeným nárustem velikosti výsledného obrázku. Původní obrázek zabíral 9816 bajtů, u *tresholdu* = 0.85 zabíral obrázek 24g, který je téměř totožný, pouze 2970 bajtů. U *tresholdu* = 0.0009 za použití nižšího počtu shluků zabíral

Tabulka 28: Velikosti obrázků v bajtech po shlukování z obrázku 25

Počet shluků	Počet vl. vektorů	Velikost
Originál	Originál	9 816 bajtů
2	1	687 bajtů
5	5	1428 bajtů
10	10	1848 bajtů
30	15	2647 bajtů
30	30	2662 bajtů
50	50	3070 bajtů
100	1	4218 bajtů
100	100	3621 bajtů
500	1	6299 bajtů
1000	1000	6985 bajtů
1000	1	7593 bajtů
2000	1	9256 bajtů
3000	1	10327 bajtů

obrázek (25f) 2662 bajtů.

Jako nejpřesnější algoritmus byl opět zvolen algoritmus Shi a Malika, kde byly objekty rozpoznatelné i při použití malého počtu shluků a uspokojivé výsledky byly vypočítány i za použití jednoho vlastního vektoru. Normovaný algoritmus, i u nízkého parametru *treshold*, detekoval pouze část obrázku a nedosahoval uspokojivých výsledků. Matice pro algoritmus Ng, Jordana Weisse zabíraly v paměti méně než pro algoritmus Shi a Malika a výpočet byl opět několikrát rychlejší. Přesnost však nastala až u větších počtu shluků, kde narůstalo jak vytížení paměti počítače, tak velikost výsledného obrázku, kde srovnatelný obrázek měl více jak dvakrát větší velikost než za použití algoritmu Shi a Malika.

7 Závěr

V dnešní době, kdy je celý svět propojen internetem, je obrovský nárůst dat na každodenním pořádku. Většina přístrojů je ve výrobních linkách připojena k internetu, a jsou schopny každou vteřinu sbírat velká množství dat, která je možno analyzovat. Analýza dat má význam i v jiných odvětvích, například v medicíně, kde jsme v dnešní době schopni na základě DNA analyzovat různé problémy a tím například ušetřit jak náklady na léky, tak zdraví pacientů. V medicíně je častá vizualizace dat pomocí segmentace obrazu.

Cílem této diplomové práce bylo prozkoumat metody využívající spektrálních rozkladů tří Laplaceových matic a na základě tří odlišných algoritmů analyzovat a vyhodnocovat různé problémy. První a stěžejní část experimentů se zabývala analýzou datových kolekcí, které obsahují neorientované vážené grafy. Z literatury převzaté algoritmy vytvářely k shluků na základě k vlastních vektorů Laplaceovy matice. Jedním z experimentů bylo zjistit, zda má vliv měnit počty vlastních vektorů a případně tak redukovat dimenzi problému. Na základě zhruba 120 000 naměřených výsledků bylo zjištěno, že u dvou algoritmů má smysl redukovat dimenzi, protože nejlepší výsledky vycházely při použití nižšího počtu vlastních vektorů než shluků.

Abychom si byli jisti, který výsledek je nejlepší, byly naimplementovány metody měřící kvalitu shlukování. Patří sem modularita, konduktance, cut ratio, silhouette index a dunnův index. Na základě těchto metrik jsme mohli určit, že nejpřesnějším algoritmem byl normovaný algoritmus pánů Shi a Malika, který používá Laplaceovu matici L_{rw}. Jedinou nevýhodou je jeho pomalejší výpočet vlastních vektorů z nesymetrické matice. Druhým nejpřesnějším algoritmem byl normovaný algoritmus Ng, Jordana a Weisse (L_{sym}), který jako jediný dosahoval lepších výsledků s vyšším počtem vlastních vektorů než shluků. Výpočetní časy pro tento algoritmus byly rychlejší a proto bychom ho spíše doporučili, i na úkor nepřesností, pro analýzu rozsáhlejších problémů. Výpočetní časy nenormovaného algoritmu (L) byly nejrychlejší, avšak algoritmus byl nejméně přesný. O měřících metrikách můžeme jednoznačně prohlásit, že neexistuje „nejlepší“ měřící metrika pro shlukování na grafech. Nelze se tedy spoléhat pouze na jeden způsob měření. Což bylo ověřeno dalším experimentem, který porovnával rozložení vrcholů grafů do shluků na základě výpočtu modularity vizualizačním programem Gephi. Metriky jsou však důležité při analýze grafů, převážně těch větších, u kterých se nelze vizuálně přesvědčit o výsledku shlukování.

Další část experimentů probíhala na barevných obrázcích. Segmentace obrazu pomocí spektrálních algoritmů pracuje s každým pixelem jako s vrcholem grafu a váhy hran značí podobnost mezi pixely. Díky této definici je potřeba brát v potaz, že i malé obrázky jsou velké datové kolekce, kde při použití obrázků o velikosti 100x100 se musíme vypořádat s 10^4 vrcholů. Podobnost mezi pixely byla počítána tzv. „gaussovským kernelem“, u kterého bylo experimentováno s parametrem *sigma*. Jakmile byla nalezena jeho nejvhodnější hodnota, experimentovalo se s parametrem *treshold*, díky kterému můžeme konstruovat řídkou matici zanedbáním nízkých podobností mezi pixely. Čím řidší matice byla, tím byl rychlejší čas výpočtu a nižší náročnost na paměť počítače, avšak za cenu nepřesností. Nejlépe opět vycházel algoritmus Shi a Malika, který,

na úkor výpočetního času, nejen nejvíce redukoval velikost původního obrázku, ale i jeho objekty byly nejlépe rozpoznatelné. Nenormovaný algoritmus v této problematice vůbec neuspěl, protože nebyl zdárně schopen rozpoznávat objekty, ani za použití vysokého počtu shluků. Normovaný algoritmus Ng, Jordana a Weisse byl opět rychlejší, ale přesnost se začala objevovat až u použití většího počtu shluků a s tím souvisejícím nárůstem velikosti výsledného obrázku.

Možná vylepšení této práce mohou spadat do problematiky paralelizace výpočtů algoritmů a jejich urychlení. Další možným vylepšením aplikace může být možnost analyzovat více druhů souborů, ve kterých jsou uloženy grafy. Úsporou času může být konstrukce matice podobnosti efektivnějším způsobem, než porovnávat každou dvojici pixelů obrázku.

Jako hlavní osobní přínos z tvorby diplomové práce hodnotím rozšíření vědomostí z oblasti analýzy dat. Bylo mi umožněno přemýšlet nad různými problémy novým pohledem a lépe se zorientovat v dané problematice. Dalším neméně důležitý osobní přínos je zlepšení programátorských dovedností. V budoucnu bych se rád věnoval analýze pohybů cen na měnových a akciových trzích.

Literatura

[1] LUXBURG, Ulrike von. *A Tutorial on Spectral Clustering.* Tübingen: Max Planck Institute for Biological Cybernetics, 2007, str. 32

[2] NG, Andrew Y., JORDAN, Michael I., WEISS, Yair *On Spectral Clustering: Analysis and an algorithm.* ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, MIT Press, 2001, str. 849–856

[3] FIEDLER, Miroslav. *Algebraic connectivity of graphs.* Praha: Czechoslovak Mathematical Journal, 1973, str. 9, 298–305

[4] SAAD, Yousef. *Numerical Methods for Large Eigenvalue Problems.* Manchester: Manchester University Press, No. Second Edition, 2011, str. 285

[5] ALMEIDA, Hélio, GUEDES, Dorgival, MEIRA, Wagner Jr., ZAKI, Mohammed J. *Is there a best quality metric for graph clusters?* MG Brasil, Universidade Federal de Minas Gerais, NY USA Rensselaer Polytechnic Institute, 2011, str. 16

[6] WAGNER, D., WAGNER, F. *Between mincut and graph bisection.* Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS), Springer, 1993, str. 744–750,

[7] LESKOVEC, Jure, YANG, Jaewon. *Defining and Evaluating Network Communities based on Ground-truth.* Stanford, Stanford University, 2012, str. 10

[8] SAITTA, Sandro, RAPHAEL, Benny, SMITH, Ian F.C. *A Bounded Index for Cluster Validity.* Switzerland, Lausanne, 2007, str. 14

[9] BRANDES, Ulrik, DELLING, Daniel, GAERTLER, Marco, GORKE, R., HOEFER, M., NIKOLOSKI, Z., WAGNER, D. *On Modularity Clustering.* IEEE Transactions on Knowledge & Data Engineering, vol.20, no. 2, 2007, str. 15

[10] WAGNER, F., Stoer, M. *A simple min-cut algorithm.* J. ACM, 44(4), 1997, str. 585-591

[11] SONG, Y., CHEN, W.-Y., BAI, H., LIN, C.-J., CHANG, E. Y. *Parallel Spectral Clustering.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.33, no. 3, March 2011, str. 568-586

[12] YAN, D., HUANG, L., JORDAN, M. I. *Fast Approximate Spectral Clustering.* 15th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), 2009

[13] SHI, J., MALIK, J. *Normalized Cuts and Image Segmentation.* IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.22, no. 8, 2000, str. 17

[14] TUNG, F., WONG, A., CLAUSI, A. *Enabling scalable spectral clustering for image segmentation.* Pattern Recognition 43, Elsevier Ltd., 2010, str. 8

[15] BACH, F. R., JORDAN, M.I. *Spectral clustering for speech separation, in Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods.* John Wiley & Sons, Chichester, 2009

[16] *Supported Graph Formats. Gephi.* [online]. 4.2.2016 [cit. 2016-02-04]. Dostupné z: https://gephi.org/users/supported-graph-formats/

[17] *Gephi Requirements. Gephi.* [online]. 4.2.2016 [cit. 2016-02-04]. Dostupné z: https://gephi.org/users/requirements/

[18] *GraphViz C# Wrapper.* [online]. 5.2.2016 [cit. 2016-02-05]. Dostupné z: https://graphviz.codeplex.com/

[19] *GraphML.* [online]. 5.2.2016 [cit. 2016-02-05]. Dostupné z: http://graphml.graphdrawing.org/

[20] *Math.NET Numerics Matrix. Math.NET Numerics.* [online]. 26.2.2016 [cit. 2016-02-26]. Dostupné z: http://numerics.mathdotnet.com/Matrix.html

[21] *Alglib.* [online]. 26.2.2016 [cit. 2016-02-26]. Dostupné z: http://www.alglib.net/

[22] *Right Eigenvector. Mathworld Wolfram.* [online]. 26.2.2016 [cit. 2016-02-26]. Dostupné z: http://mathworld.wolfram.com/RightEigenvector.html

[23] *Fast unfolding of communities in large networks, in Journal of Statistical Mechanics: Theory and Experiment* Blondel, Vincent D., GUILLANE, Jean-Loup,LAMBIOTTE, Renaud,LEFEBRE, Etienne, 2008 (10), P1000

[24] *Stanford Large Network Dataset Collection.* Jure Leskovec. Snap Stanford. [online]. 6.3.2016 [cit. 2016-03-06]. Dostupné z: http://snap.stanford.edu/data/

[25] *The Koblenz Network Collection.* Konect Koblenz University. [online]. 6.3.2016 [cit. 2016-03-06]. Dostupné z: http://konect.uni-koblenz.de/

[26] *Train Bombing.* The Koblenz Network Collection. Konect Koblenz University. [online]. 29.3.2016 [cit. 2016-03-29]. Dostupné z: http://konect.uni-koblenz.de/networks/moreno_train

[27] *Statistický jazyk a program R.* The R Project for Statistical Computing [online]. 8.3.2016 [cit. 2016-03-08]. Dostupné z: https://cran.r-project.org/

[28] *Ckmeans.1d.dp: Optimal k-Means Clustering for One-Dimensional Data.* The R Project for Statistical Computing [online]. 8.3.2016 [cit. 2016-03-08]. Dostupné z: https://cran.r-project.org/web/packages/Ckmeans.1d.dp/index.html

[29] DOSTÁL, Zdeněk, VONDRÁK, Vít *Lineární algebra.* Matematika pro inženýry 21. století, Vysoká škola báňská – Technická univerzita Ostrava a Západo-česká univerzita v Plzni, 2011, str. 212

[30] BROWN,R. E., STON, M. A. *On the use of polynomial series with the Rayleigh-Ritz method.* Branch of Aeronautical Engineering School of Mechanical Engineering Universityof the Witwatersrand,PO. WITS, 2050, WitwatersTand, South Africa, 1997, str. 197-196

[31] ŘEZÁNKOVÁ, Hana, HÚSEK, Dušan, SNÁŠEL, Václav *Shluková analýza dat.* 2., rozš. vyd. Praha: Professional Publishing, 2009. ISBN 978-80-86946-81-8.

[32] ALZATE, C., SUYKENS, J. A. K. *Image Segmentation using a Weighted Kernel PCA Approach to Spectral Clustering.* Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on, str. 208-213

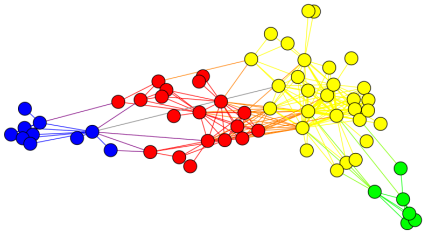
[33] *GraphViz.* GraphViz. [online]. 29.3.2016 [cit. 2016-03-29]. Dostupné z: http://graphviz.org/

[34] ARTHUR, David, VASSILVITSKII, Sergei *K-means++: the advantages of careful seeding.* In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, str. 1027-1035

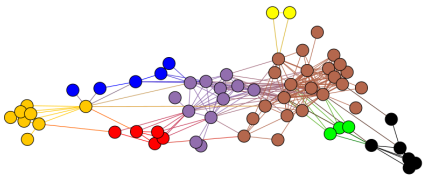
A Experimenty nad grafovými datovými kolekcemi

V této části budou uvedeny tabulky a vizualizace grafů, které se nevešly do základního textu diplomové práce.

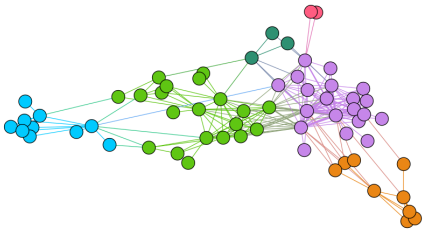
A.1 Datová kolekce obsahující teroristy



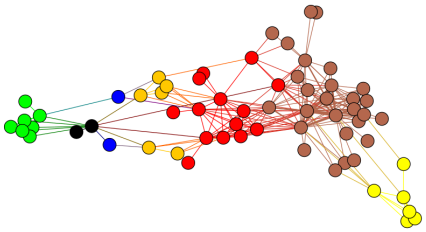
Obrázek 26: Teroristi - Normovaný algoritmus Shi a Malika. 4 shluky a 6 vlastních vektorů.



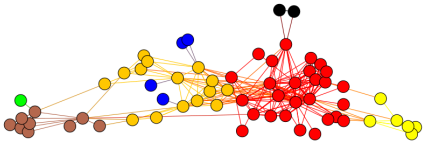
Obrázek 27: Teroristi - Normovaný algoritmus Shi a Malika. 8 shluků a 6 vlastních vektorů.



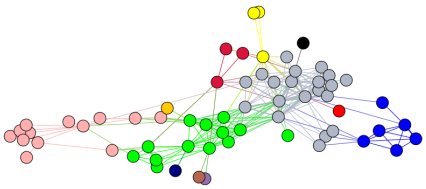
Obrázek 28: Teroristi a vrcholy rozděleny do 6 shluků na základě výpočtu modularity pomocí programu Gephi.



Obrázek 29: Teroristi - Normovaný algoritmus Shi a Malika. 7 shluků a 1 vlastní vektor.



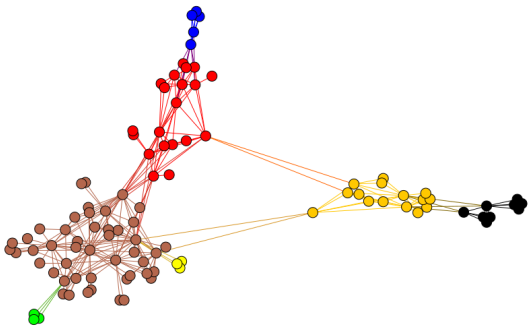
Obrázek 30: Teroristi - Nenormovaný algoritmus 7 shluků a 3 vlastní vektory.



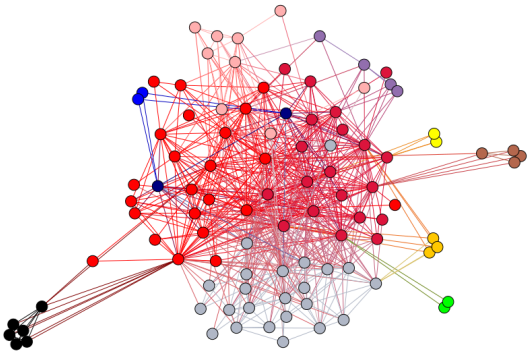
Obrázek 31: Teroristi - Normovaný algoritmus Ng, Jordana a Weisse. 12 shluků a 13 vlastních vektor;.

A.2 Různě rozsáhlé syntetické sítě

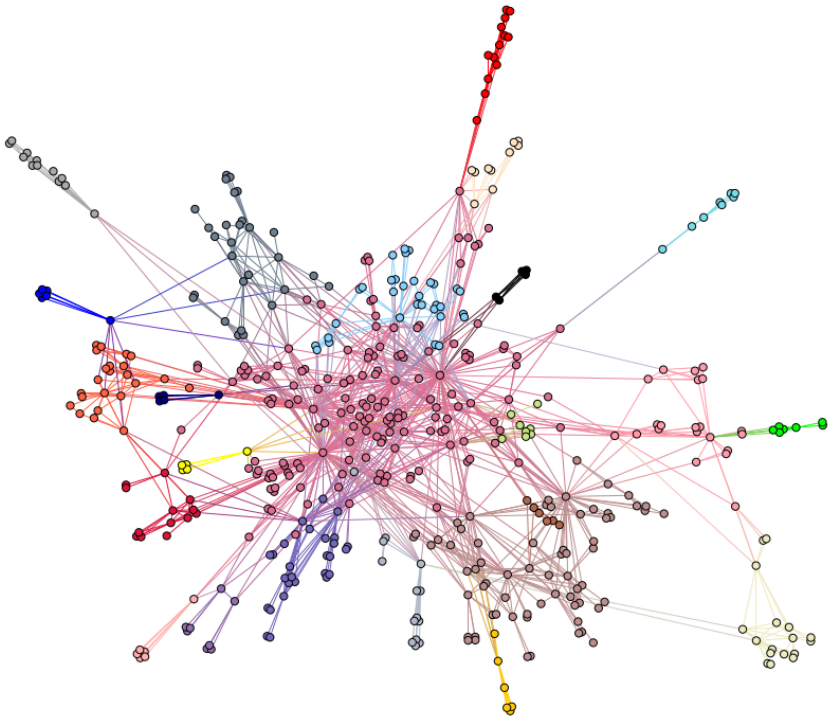
V této kapitole přílohy se nachází vizualizace programem Gephi různě rozsáhlých syntetických sítí.



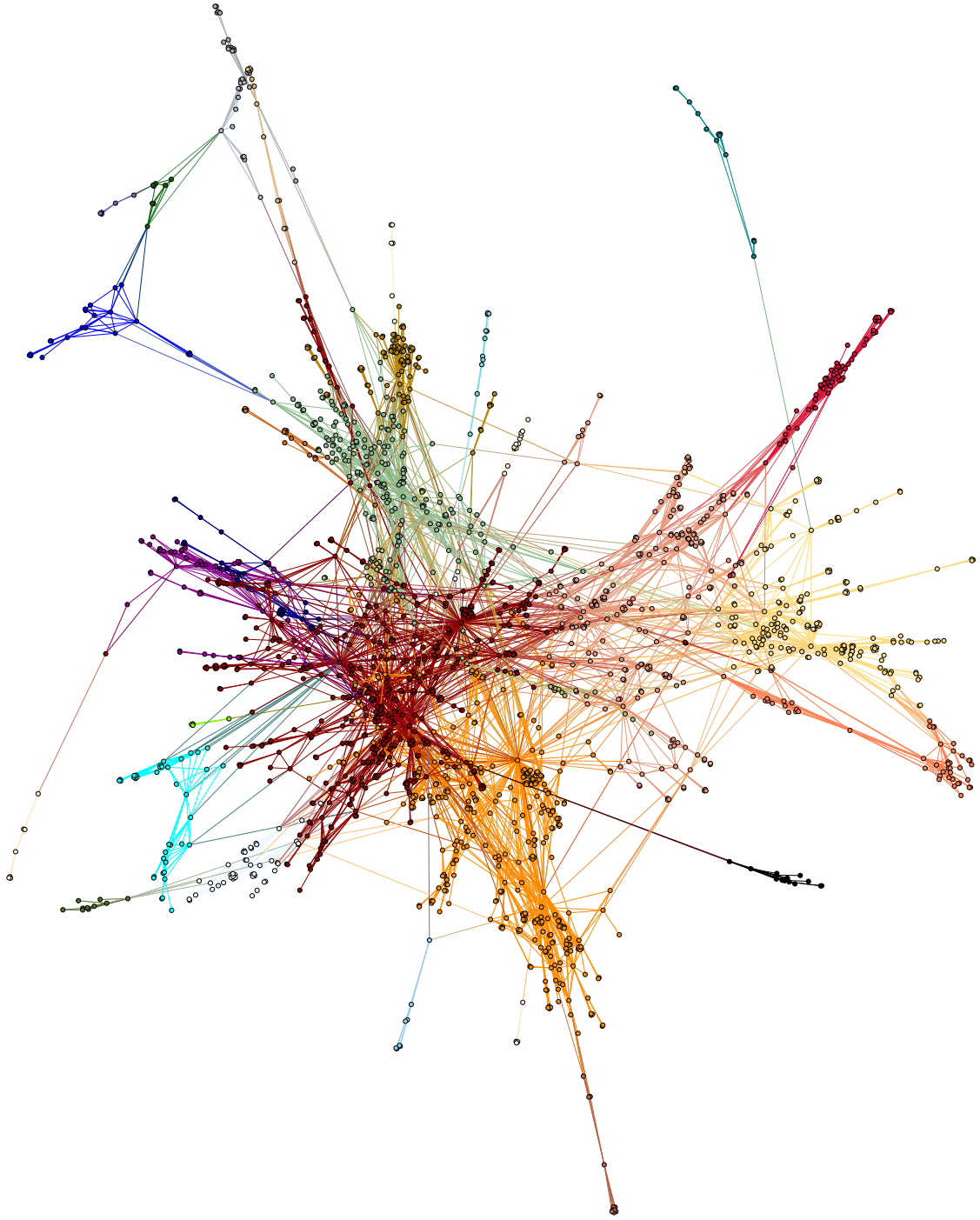
Obrázek 32: Datová kolekce D_100_318 - Normovaný algoritmus Shi a Malika. 7 shluků a 6 vlastních vektorů.



Obrázek 33: Datová kolekce D_100_685 - Normovaný algoritmus Shi a Malika. 12 shluků a 8 vlastních vektorů.



Obrázek 34: Datová kolekce **D_501_1502** - Normovaný algoritmus Shi a Malika. 24 shluků a 23 vlastních vektorů.



Obrázek 35: Datová kolekce **D__2000__5700** - Normovaný algoritmus Shi a Malika. 31 shluků a 26 vlastních vektorů.

Obrázek 36: Datová kolekce D_2000_12306 - Normovaný algoritmus Shi a Malika. 81 shluků a 48 vlastních vektorů.

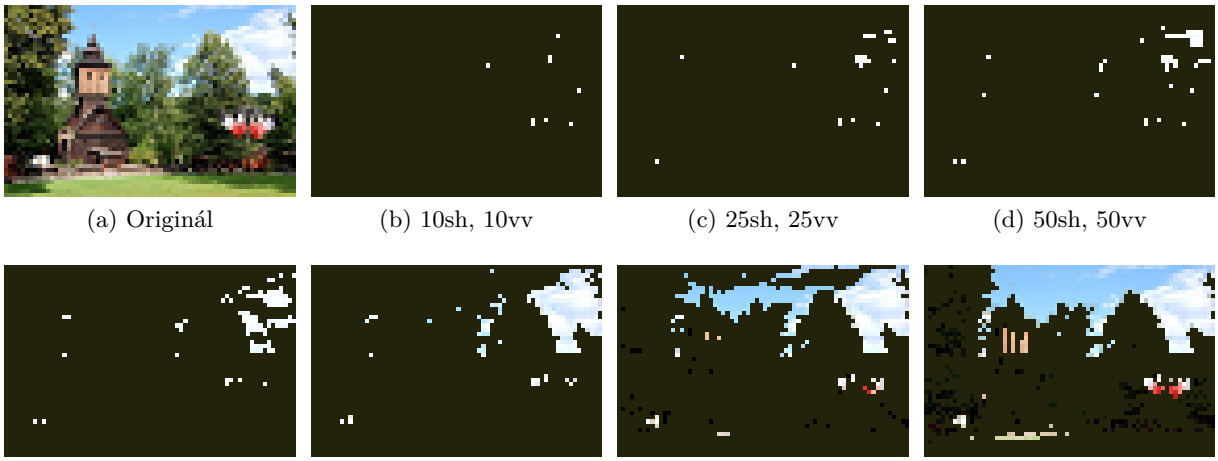
B Experimenty nad obrázky

V této části přílohy se nachází experimenty s obrázky, které se nevešly do hlavní části diplomové práce.

B.1 Segmentace obrazu na obrázku rožnovského skanzenu

B.1.1 Nenormovaný algoritmus

Na obrázku 37 můžeme vidět segmentaci obrazu pomocí nenormovaného algoritmu s různými počty shluků s parametry *sigma* = 0.0009 a *treshold* = 0.0009. Pro tyto obrázky je zde tabulka 29, kde vidíme výsledky. Matice zabíraly v paměti 747 - 898 MB.



(a) Originál (b) 10sh, 10vv (c) 25sh, 25vv (d) 50sh, 50vv

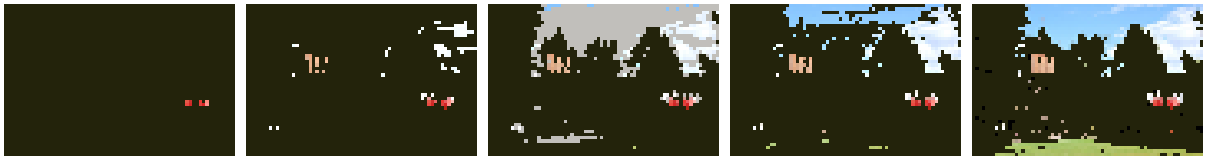
(e) 100sh, 100vv (f) 250sh, 250vv (g) 500sh, 500vv (h) 1000sh, 1000vv

Obrázek 37: Skanzen po použití nenormovaného algoritmu. Parametry *sigma* = 0.0009 a *treshold* = 0.0009.

Tabulka 29: Velikosti obrázků v bajtech po shlukování z obrázku 37

Počet shluků	Počet vl. vektorů	Velikost
Originál	Originál	9816 bajtů
10	10	265 bajtů
25	25	305 bajtů
50	50	397 bajtů
100	100	550 bajtů
250	250	1026 bajtů
500	500	1747 bajtů
1000	1000	3320 bajtů

Na obrázku 38 vidíme vizualizaci za použití parametrů *sigma* = 0.0009 a *treshold* = 0.85. Tabulka 30 zobrazuje naměřené časy a velikosti výsledných obrázků. Matice pro tyto výpočty zabíraly v paměti od 450 MB po 592 MB.



(a) 10sh, 10vv (b) 100sh, 100vv (c) 200sh, 200vv (d) 500sh, 500vv (e) 1000sh, 1000vv

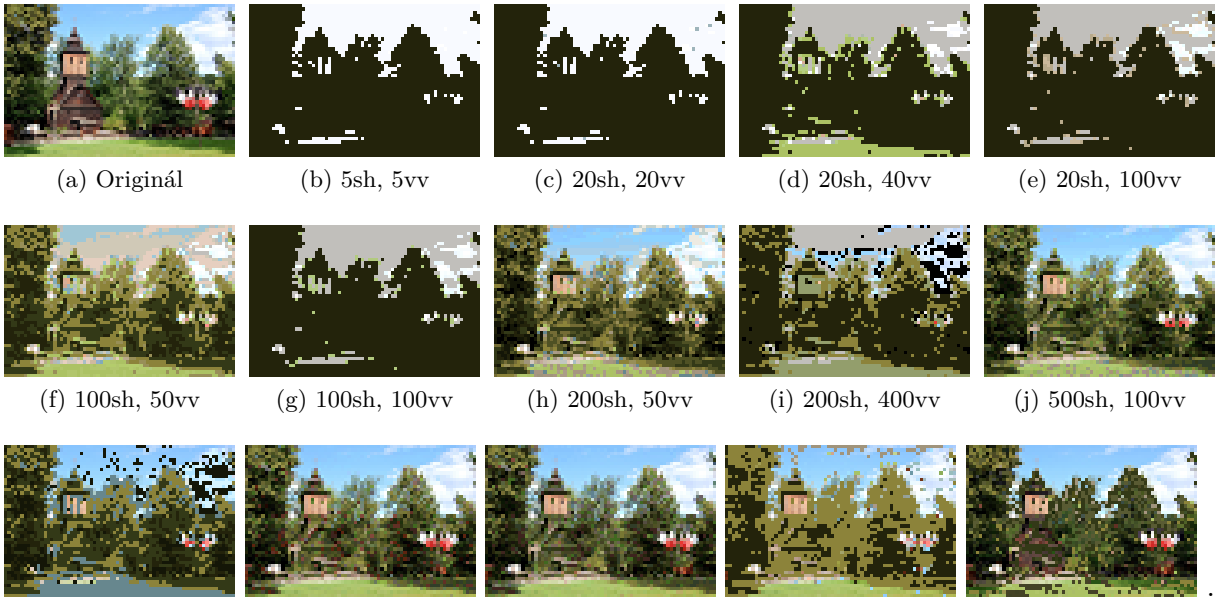
Obrázek 38: Skanzen po použití nenormovaného algoritmu. Parametry *sigma* = 0.0009 a *treshold* = 0.85.

Tabulka 30: Velikosti obrázků v bajtech po shlukování z obrázku 38

Počet shluků	Počet vl. vektorů	Velikost
Originál	Originál	9816 bajtů
10	10	295 bajtů
100	100	654 bajtů
200	200	1235 bajtů
500	500	1762 bajtů
1000	1000	3187 bajtů

B.1.2 Normovaný algoritmu Ng, Jordana a Weisse

Na obrázku 39 lze vidět segmentaci obrazu pomocí nenormovaného algoritmu s různými počty shluků s parametry *sigma* = 0.0009 a *treshold* = 0.0009.



(a) Originál (b) 5sh, 5vv (c) 20sh, 20vv (d) 20sh, 40vv (e) 20sh, 100vv

(f) 100sh, 50vv (g) 100sh, 100vv (h) 200sh, 50vv (i) 200sh, 400vv (j) 500sh, 100vv

(k) 500sh, 1000vv (l) 1000sh, 100vv (m) 1000sh, 500vv (n) 1000sh, 2000vv (o) 2000sh, 3000vv

Obrázek 39: Skanzen po použití normovaného algoritmu Ng, Jordana a Weisse. Parametry *sigma* = 0.0009 a *treshold* = 0.0009.

Matice pro tento obrázek zabíraly v paměti od 690 po 1140 MB. Tabulka 31 zobrazuje naměřené hodnoty.

Tabulka 31: Velikosti obrázků pro obrázek 39 rožnovského skanzenu v bajtech po shlukování

Počet shluků	Počet vl. vektorů	Velikost
Originál	Originál	9816 bajtů
5	5	562 bajtů
5	10	562 bajtů
10	10	562 bajtů
20	40	982 bajtů
20	100	877 bajtů
50	50	716 bajtů
50	250	1115 bajtů
100	50	2394 bajtů
200	50	3432 bajtů
500	100	5078 bajtů
500	1000	3092 bajtů
1000	100	6895 bajtů
1000	500	6364 bajtů
1000	2000	4307 bajtů
2000	3000	7745 bajtů

C Obsah přiloženého CD

Pro detailnější informace jsou v některých složkách vytvořeny textové soubory s popisem.

1. Testovací aplikace včetně zdrojových kódů
2. Datové soubory obsahující grafy
   (a) Databáze s naměřenými výsledky ve formátu mdf
   (b) Datové kolekce, který byly analyzovány v této práci
   (c) Datasety vizualizované programem Geph
   (d) Ostatní soubory, které nejsou v textu práce
3. Obrázky
   (a) Výsledky experimentů nad obrázkem skanzenu
   (b) Výsledky experimentů parametru sigma
   (c) Výsledky experimentů parametru treshold
   (d) Ostatní experimenty, které nejsou v textu práce